OBJECT WINDOWS® C++

REFERENZ-HANDBUCH

BORLAND

ObjectWindows für C++

ObjectWindows für C++

Referenzhandbuch

BORLAND

Copyright © 1991 by Borland International, INC.
Jede Vervielfältigung dieses Handbuches sowie des ObjectWindows Softwareprogrammes wird strafrechtlich verfolgt. Die Rechte an der Dokumentation und die Rechte an dem Softwareprogramm ObjectWindows liegen bei Borland International.

Der rechtmäßige Erwerb der Programmdisketten und der Handbücher erlaubt die Nutzung der Programme analog der Benutzung eines Buches. Entsprechend der Unmöglichkeit, daß ein Buch zugleich an verschiedenen Orten von mehreren Personen gelesen wird, darf das Softwareprogramm ObjectWindows nicht gleichzeitig von verschiedenen Personen an verschiedenen Orten und auf verschiedenen Geräten benutzt werden. Diskettenkopien dürfen lediglich zum Zweck der Datensicherung angefertigt werden.

Sie (der Lizenznehmer von ObjectWindows) dürfen eigene Programme, die Sie mit dem Borland C++-Sprachcompiler geschrieben und compiliert haben, ohne Einschränkung oder Verpfichtung zu zusätzlichen Lizenzgebühren verwenden, weitergeben oder verkaufen, solange diese Programme einen Copyright-Vermerk enthalten. Unter Copyright-Vermerk verstehen wir entweder Ihren eigenen Copyright-Vermerk oder gegebenenfalls einen Hinweis wie »Created using ObjectWindows by Borland International 1991«.

Die Beispielprogramme von ObjectWindows, die auf den Originaldisketten mitgeliefert werden, demonstrieren die verschiedenen Eigenschaften von Obejct-Windows. Sie dienen ausschließlich zu Lehrzwecken. Sie dürfen diese Beispielprogramme editieren oder für den eigenen Gebrauch modifizieren. Sie dürfen diese Programme jedoch nicht als Ganzes oder in Teilen in compilierter Form oder als Quelltext weitergeben oder verkaufen. Sie dürfen die Routinen aus den Beispielprogrammen in eigenen Anwendungsprogrammen verwenden, solange die resultierenden Programme nicht einem Bespielprogramm oder einem Teil eines Beispielprogrammes in Erscheinung oder Funktion gleichen.

Einschränkung der Gewährleistung
Es wird keine Garantie für die Richtigkeit des Inhaltes dieses Handbuches übernommen. Da sich Fehler, trotz aller Bemühungen, nie vollständig vermeiden lassen, sind wir für Hinweise jederzeit dankbar.

Herausgeber
Borland GmbH
Gautinger Straße 10
Postfach
8130 Starnberg
Telefon 0 81 51 / 2 63 - 0
Telefax 0 81 51 / 149 09

Umschlag und typografische Konzeption: Rudolf Paulus Gorbach

Übertragung ins Deutsche: Peter Baier, Roland Konstanty, Waltraud Kopelent

Typographische Konzeption des Innenteils: Rudolf Paulus Gorbach

Grafiken: Sid Gastl

Satz: Hanna Bergbauer

PostScript-Belichtung: CD Computer & Dokumentations GmbH, Aalen

Druck und Bindung: Kösel GmbH, Kempten

2., unveränderte Auflage, Mai 1992

Inhalt

1. Windows-Funktionen 1

GDI-Funktionen 1
Abbildungsfunktionen 1
Bitmapfunktionen 2
Clipping-Funktionen 3
Drucker-Escape-Funktion 4
Druckersteuerfunktionen 4
Ellipsen- und Vieleckfunktionen 5
Farbpalettenfunktionen 5
Gerätekontextfunktionen 6
Geräteunabhängige Bitmapfunktionen 7
Koordinatenfunktionen 8
Metafile-Funktionen 8
Rechteckfunktionen 9
Regionfunktionen 10
Schriftfunktionen 11
Textfunktionen 11
Umgebungsfunktionen 12
Zeichnungsattributfunktionen 12
Zeichenwerkzeugfunktionen 13
Funktionen der Windows-API 14
Linienausgabefunktionen 14
Anwendungsausführungsfunktionen 14
Atomverwaltungsfunktionen 15
Betriebssystem-Interruptfunktionen 16
Dateiein-/ausgabefunktionen 16
Debugging-Funktionen 17
Dienstmakros und -funktionen 17
Initialisierungsdateifunktionen 18
Klangfunktionen 19
Modulverwaltungsfunktionen 20
Ressourcenverwaltungsfunktionen 21
Schnittstellenfunktionen 21
Segmentfunktionen 22
Speicherverwaltungsfunktionen 23
Stringmanipulationsfunktionen 26
Taskfunktionen 27
Schnittstellenfunktionen der Fenster-Verwaltung (WMI) 28
Anzeige- und Verschiebefunktionen 28

Bildlauffunktionen 29
Botschaftsfunktionen 30
Caret-Funktionen 31
Dialogfensterfunktionen 31
Eigenschaftsfunktionen 33
Eingabefunktionen 34
Fehlerfunktionen 35
Fenstererstellungsfunktionen 35
Filterverwaltungsfunktionen 36
Hardwarefunktionen 37
Informationsfunktionen 37
Mauszeigerfunktionen 38
Menüfunktionen 39
Systemfunktionen 40
Zeichenfunktionen 40
Zwischenablagefunktionen 41
Programmabbruchcodes 42
Referenz aller Windows-Funktionen 44

2. Windows-Botschaften 417

3. Windows-Typen 513

4. Windows-Stile und -Konstanten 547

Index 627

1. Windows-Funktionen

Dieses Kapitel enthält eine inhaltlich geordnete Übersicht über alle Windows-Funktionen mit einer Kurzbeschreibung der jeweiligen Funktion. Auf diese Zusammenstellung folgt dann eine ausführliche Beschreibung aller Funktionen in alphabetischer Reihenfolge.

GDI-Funktionen

Die Funktionen der GDI (grafische Geräteschnittstelle) sind zuständig für die Ausgabe von Graphiken auf den Bildschirm oder den Drucker. Diese Funktionen werden im folgenden beschrieben.

Abbildungsfunktionen

Abbildungsfunktionen ändern GDI-Abbildungsmodi und liefern Informationen über diese. Um die Geräteunabhängigkeit aufrechtzuerhalten, erzeugt die GDI eine Ausgabe in einem virtuellen Raum und bildet das Resultat auf dem Bildschirm ab. Der Abbildungsmodus definiert das Verhältnis zwischen Einheiten im virtuellen Raum und Pixeln auf einem Gerät. Die folgende Liste beschreibt kurz jede Abbildungsfunktion:

GetMapMode	Liefert den aktuellen Abbildungsmodus.
GetViewportExt	Liefert die Grafikfensterabmessungen eines Gerätekontextes.
GetViewportOrg	Liefert den Ursprung des Grafikfensters eines Gerätekontextes.
GetWindowExt	Liefert die Fensterabmessungen eines Gerätekontextes.
GetWindowOrg	Liefert den Fensterursprung eines Gerätekontextes.
OffsetViewportOrg	Verändert einen Grafikfensterursprung.

OffsetWindowOrg	Verändert einen Fensterursprung.
ScaleViewportExt	Verändert die Grafikfensterabmessungen.
ScaleWindowExt	Verändert die Fensterabmessungen.
SetMapMode	Setzt den Abbildungsmodus eines angegebenen Gerätekontextes.
SetViewportExt	Setzt die Grafikfensterabmessungen eines Gerätekontextes.
SetViewportOrg	Setzt den Grafikfensterursprung eines Gerätekontextes.
SetWindowExt	Setzt die Fensterabmessungen eines Gerätekontextes.
SetWindowOrg	Setzt den Fensterursprung eines Gerätekontextes.

Es gibt acht verschiedene Abbildungsmodi: *MM_ANISOTROPIC*, *MM_HIENGLISH*, *MM_HIMETRIC*, *MM_ISOTROPIC*, *MM_LOENGLISH*, *MM_LOMETRIC*, *MM_TEXT*, *MM_TWIPS*. Jeder Modus erfüllt in einer Windows-Anwendung einen besonderen Zweck. Die Tabelle »GDI-Abbildungsmodi« in Kapitel 4 »Windows-Stile und -Konstanten« erläutert diese Abbildungsmodi.

Bitmap-Funktionen

Bitmap-Funktionen stellen Bitmaps dar. Ein Bitmap ist eine Matrix aus Speicherbits, die, wenn sie zu einem Gerät kopiert wird, die Farbe und das Muster einer entsprechenden Matrix aus Pixeln auf der Bildschirmoberfläche des Gerätes definieren. Bitmaps sind in Anwendungen nützlich, die das Zeichnen, die graphische Darstellung oder die Textverarbeitung zur Aufgabe haben. Sie ermöglichen die Vorbereitung von Bildern im Speicher, um sie dann schnell auf den Bildschirm zu kopieren. Die folgende Liste beschreibt jede Bitmap-Funktion:

BitBlt	Kopiert ein Bitmap von einem Quell- zu einem Zielgerät.
CreateBitmap	Erzeugt ein Bitmap.
CreateBitmapIndirect	Erzeugt ein in einer Datenstruktur definiertes Bitmap.
CreateCompatibleBitmap	Erzeugt ein Bitmap, das zu einem bestimmten Gerät kompatibel ist.
CreateDiscardableBitmap	Erzeugt ein Bitmap, dem das Attribut »discardable« zugeordnet ist (d.h. dessen Speicherbereich verworfen werden kann), das zu einem bestimmten Gerät kompatibel ist.

ExtFloodFill	Füllt die Bildschirmoberfläche innerhalb einer Grenze oder eines Bereichs mit einer vorgegebenen Farbe aus.
FloodFill	Füllt die Bildschirmoberfläche innerhalb einer Grenze aus.
GetBitmapBits	Liefert die Bits für ein bestimmtes Bitmap aus dem Speicher.
GetBitmapDimension	Liefert die Abmessungen eines Bitmaps.
GetPixel	Liefert den RGB-Wert eines Pixel.
LoadBitmap	Lädt ein Bitmap aus einer Ressourcendatei.
PatBlt	Erzeugt ein Bitmuster.
SetBitmapBits	Setzt die Bits eines Bitmaps.
SetBitmapDimension	Setzt Höhe und Breite eines Bitmaps.
SetPixel	Setzt den RGB-Wert eines Pixel.
StretchBlt	Kopiert ein Bitmap von einem Quell- zu einem Zielgerät (und staucht oder streckt es bei Bedarf).

Clipping-Funktionen

Clipping-Funktionen erzeugen, überprüfen und verändern Clipping-Regionen. Eine Clipping-Region ist der Teil des Client-Bereiches eines Fensters, in dem die GDI Ausgaben erzeugt. Jede Ausgabe, die an den Teil des Client-Bereiches gesendet wird, der außerhalb der Clipping-Region liegt, wird nicht sichtbar. Clipping-Regionen sind in allen Windows-Anwendungen nützlich, die einen Teil des Client-Bereiches speichern müssen und gleichzeitig Ausgabe an einen anderen senden müssen. Die folgende Liste beschreibt jede Clipping-Funktion:

ExcludeClipRect	Schließt ein Rechteck aus der Clipping-Region aus.
GetClipBox	Kopiert die Abmessungen eines einschließenden Rechtecks.
IntersectClipRect	Bildet die Schnittmenge einer Clipping-Region und eines Rechtecks.
OffsetClipRgn	Verschiebt eine Clipping-Region.
PtVisible	Ermittelt, ob ein Punkt innerhalb einer Region liegt.
RectVisible	Bestimmt, ob ein Teil eines Rechtecks in einer Region liegt.
SelectClipRgn	Wählt eine Clipping-Region aus.

Drucker-Escape-Funktion

Die Drucker-Escape-Funktion erlaubt es einer Anwendung, auf die Fähigkeiten eines bestimmten Gerätes zuzugreifen, die nicht direkt über die GDI zugänglich sind. Der Integer-Parameter *Escape* dieser Funktion bestimmt die auszuführende Funktion. Wenn eine Anwendung *Escape* für einen Druckergerätekontext aufruft, steuern die Sonderfunktionen den Fluß der Druckerausgaben von Windows-Anwendungen, liefern Informationen über einen Drucker und ändern die Einstellung eines Druckers.

Druckersteuerfunktionen

Dies ist eine alphabetische Liste der spezifischen Drucker-Escape-Funktionen von Windows. Diese Funktionen ermöglichen es Anwendungen, die Fähigkeiten eines bestimmten Ausgabegerätes zu nutzen, die nicht direkt über die GDI abrufbar sind. Die Escape-Funktionen werden von einer Anwendung aufgerufen, von Windows übersetzt und dann an den Druckertreiber gesendet.

ABORTDOC
BANDINFO
BEGIN_PATH
CLIP_TO_PATH
DEVICEDATA
DRAFTMODE
DRAWPATTERNRECT
ENABLEDUPLEX
ENABLEPAIRKERNING
ENABLERELATIVEWIDTHS
ENDDOC
END_PATH
ENUMPAPERBINS
ENUMPAPERMETRICS
EPSPRINTING
EXT_DEVICE_CAPS
EXTTEXTOUT
FLUSHOUTPUT
GETCOLORTABLE
GETEXTENDEDTEXTMETRICS
GETEXTENTTABLE
GETFACENAME
GETPAIRKERNTABLE
GETPHYSPAGESIZE
GETPRINTINGOFFSET
GETSCALINGFACTORG
GETSETPAPERBINS
GETSETPAPERMETRICS
GETSETPRINTORIENT
GETSETSCREENPARAMS
GETTECHNOLOGY
GETTRACKKERNTABLE
GETVECTORBRUSHSIZE
GETVECTORPENSIZE
MFCOMMENT
NEWFRAME
NEXTBAND
PASSTHROUGH
QUERYESCSUPPORT
RESTORE_CTM
SAVE_CTM
SELECTPAPERSOURCE
SETABORTPROC
SETALLJUSTVALUES
SET_ARC_DIRECTION
SET_BACKGROUND_COLOR
SET_BOUNDS
SETCOLORTABLE
SETCOPYCOUNT
SETKERNTRACK
SETLINEJOIN
SETMITERLIMIT
SET_POLY_MODE
SET_SCREEN_ANGLE
SET_SPREAD
STARTDOC
TRANSFORM_CTM

Ellipsen- und Vieleckfunktionen

Ellipsen- und Vieleckfunktionen zeichnen Ellipsen und Vielecke (Polygone). Die GDI zeichnet den Umfang jedes Objekts mit dem ausgewählten Stift und füllt das Innere mit dem ausgewählten Pinsel. Diese Funktionen sind besonders nützlich in Anwendungen zum Zeichnen oder zur Diagrammdarstellung. Die folgende Liste beschreibt kurz jede Ellipsen- und Vieleckfunktion:

Chord	Zeichnet eine Sehne.
DrawFocusRect	Zeichnet ein Rechteck in dem Stil, der benutzt wird, um den Fokus anzuzeigen.
Ellipse	Zeichnet eine Ellipse.
Pie	Zeichnet ein Tortendiagramm.
Polygon	Zeichnet ein geschlossenes Vieleck.
PolyPolygon	Zeichnet eine Folge von offenen oder geschlossenen Vielecken, die mit dem Füllmodus *ALTERNATE* ausgefüllt werden.
Rectangle	Zeichnet ein Rechteck.
RoundRect	Zeichnet ein Rechteck mit abgerundeten Ecken.

Farbpaletten-funktionen

Die üblichen Farbgrafikbildschirme sind in der Lage, sehr viele Farben darzustellen. In den meisten Fällen ist jedoch die Zahl der tatsächlich gleichzeitig darstellbaren Farben eher begrenzt. So kann zum Beispiel aufgrund der Leistungsgrenzen der Hardware ein Bildschirm, der potentiell in der Lage ist, 262000 verschiedene Farben zu erzeugen, nur 256 davon gleichzeitig darstellen. In solchen Fällen verwaltet der Bildschirm (bzw. die Grafikkarte) oft eine Farbpalette: wenn eine Anwendung eine Farbe anfordert, die gegenwärtig nicht dargestellt ist, dann fügt der Bildschirm die erforderliche Farbe zur Palette hinzu. Wenn jedoch die Zahl der angeforderten Farben die Maximalzahl für das Gerät überschreitet, muß es eine vorhandene Farbe mit der angeforderten Farbe austauschen. Die Tatsache, daß die Gesamtzahl der von einem oder mehreren Fenstern angeforderten Farben die der auf dem Gerät darstellbaren häufig übersteigt, kann zu einer falschen Farbdarstellung auf dem Bildschirm führen.

Windows-Farbpaletten wirken als Puffer zwischen farbintensiven Anwendungen und dem System. Sie erlauben es einer Anwendung, so viele Farben wie nötig zu verwenden, ohne sich mit ihrer eigenen Farbdarstellung oder den von anderen Fenstern dargestellten Farben zu überschneiden. Wenn ein Fenster den Fokus hat, dann stellt Windows sicher, daß das Fenster alle erforderlichen Farben darstellt, bis hin zu der maximalen Anzahl der gleichzeitig auf dem Bildschirm verfügbaren Farben. Zusätzliche Farben werden dargestellt, indem

sie an verfügbare Farben angeglichen werden. Außerdem gleicht Windows die von inaktiven Fenstern angeforderten Farben den verfügbaren Farben so nah wie möglich an. Dies reduziert unerwünschte Farbänderungen in inaktiven Fenstern deutlich.

Die folgende Liste beschreibt kurz die Funktionen, die eine Anwendung aufruft, um Farbpaletten zu benutzen:

AnimatePalette	Ersetzt Einträge in einer virtuellen Palette; Windows bildet die neuen Einträge sofort in die Systempalette ab.
CreatePalette	Erzeugt eine virtuelle Palette.
GetNearestPaletteIndex	Liefert den Index eines virtuellen Paletteneintrags, der dem vorgegebenen RGB-Farbwert am ehesten entspricht.
GetPaletteEntries	Liefert Einträge aus einer virtuellen Palette.
GetSystemPaletteEntries	Liefert einen Bereich von Paletteneinträgen aus der Systempalette.
GetSystemPaletteUse	Bestimmt, ob eine Anwendung Zugriff auf die gesamte Systempalette hat.
RealizePalette	Bildet Einträge in einer virtuellen Palette auf die Systempalette ab.
SelectPalette	Wählt eine virtuelle Palette in einen Gerätekontext.
SetPaletteEntries	Setzt neue Paletteneinträge in einer virtuellen Palette; Windows bildet die neuen Einträge nicht auf die Systempalette ab, bevor die Anwendung die virtuelle Palette realisiert.
SetSystemPaletteUse	Ermöglicht es einer Anwendung, die gesamte Systempalette zu benutzen.
UpdateColors	Führt eine pixelweise Übersetzung der aktuellen Farbe jedes Pixels zur Systempalette aus. Dies ermöglicht es einem inaktiven Fenster, seine Farben zu korrigieren, ohne seinen Client-Bereich neu zu zeichnen.

Gerätekontext-funktionen

Gerätekontextfunktionen erzeugen und löschen Gerätekontexte (DC) und stellen sie wieder her. Ein Gerätekontext ist das Bindeglied zwischen einer Windows-Anwendung, einem Gerätetreiber und einem Ausgabegerät, wie einem Drucker oder Plotter.

Jede Windows-Anwendung kann GDI-Funktionen benutzen, um auf ein

Ausgabegerät zuzugreifen. Die GDI (grafische Geräteschnittstelle) übergibt geräteunabhängige Aufrufe von der Anwendung an den Gerätetreiber. Der Gerätetreiber übersetzt dann die Aufrufe in geräteabhängige Operationen.

Die folgende Liste beschreibt kurz jede Gerätekontextfunktion:

CreateCompatibleDC	Erzeugt einen Speichergerätekontext.
CreateDC	Erzeugt einen Gerätekontext.
CreateIC	Erzeugt einen Informationskontext.
DeleteDC	Löscht einen Gerätekontext.
GetDCOrg	Liefert den Ursprung eines Gerätekontexts.
RestoreDC	Stellt einen Gerätekontext wieder her.
SaveDC	Speichert den aktuellen Status des Gerätekontexts.

Geräteunabhängige Bitmapfunktionen

Windows bietet in der Version 3.0 eine Reihe von Funktionen, die Farbbitmaps definieren oder manipulieren, die auf jedem Gerät mit einer gegebenen Auflösung passend dargestellt werden können, ungeachtet der Methode, mit der der Bildschirm Farben im Speicher darstellt. Diese Funktionen übersetzen eine geräteunabhängige Bitmapbeschreibung in das gerätebezogene Format, das vom aktuellen Bildschirm benutzt wird. Nachfolgend eine Liste dieser Funktionen:

CreateDIBitmap	Erzeugt ein gerätespezifisches Speicherbitmap aus einer geräteunabhängigen Bitmapbeschreibung (DIB) und initialisiert (optional) Bits im Bitmap. Diese Funktion entspricht im wesentlichen *CreateBitmap.*
GetDIBits	Liefert die Bits für ein bestimmtes Bitmap in geräteunabhängigem Format aus dem Speicher. Diese Funktion entspricht im wesentlichen *GetBitmapBits.*
SetDIBits	Setzt die Bits eines Speicherbitmaps aus einem DIB. Diese Funktion entspricht im wesentlichen *SetBitmapBits.*
SetDIBitsToDevice	Setzt Bits direkt aus einem DIB auf die Bildschirmoberfläche.
StretchDIBits	Verschiebt ein geräteunabhängiges Bitmap (DIB) aus einem Quellrechteck in ein Zielrechteck und streckt oder komprimiert es (falls erforderlich).

Eine geräteunabhängige Bitmapbeschreibung besteht aus zwei Teilen:

1. Einer Datenstruktur *BITMAPINFO,* die das Format des Bitmaps vorgibt und optional eine vom Bitmap benutzte Farbtabelle enthält.

2. Einem Array von Bytes, die die Bitmap-Bitwerte enthalten.

In Abhängigkeit von den in der Bitmap-Informationsdatenstruktur enthaltenen Werten können die Bitmap-Bitwerte explizite Farbwerte (RGB) oder Indizes auf die Farbtabelle bestimmen. Außerdem kann die Farbtabelle aus Indizes auf die aktuell realisierte virtuelle Palette bestehen, statt aus expliziten Farbwerten. Beachten Sie, daß der Ursprung des Koordinatensystems für DIBs die untere linke Ecke ist, nicht die für Windows standardmäßig gültige obere linke Ecke.

Koordinaten-funktionen

Koordinatenfunktionen konvertieren Client-Koordinaten in Bildschirmkoordinaten (oder umgekehrt) und bestimmen die Position eines bestimmten Punktes. Diese Funktionen sind in grafikintensiven Anwendungen von Nutzen. Die folgende Liste beschreibt kurz jede Koordinatenfunktion:

ChildWindowFromPoint	Bestimmt, welches untergeordnete Fenster einen bestimmten Punkt enthält.
ClientToScreen	Konvertiert Client-Koordinaten in Bildschirmkoordinaten.
DPtoLP	Konvertiert Gerätepunkte (relativ zum Fensterursprung) in virtuelle Punkte.
LPtoDP	Konvertiert virtuelle Punkt in Gerätepunkte.
ScreenToClient	Konvertiert Bildschirmkoordinaten in Client-Koordinaten.
WindowFromPoint	Bestimmt, welches Fenster einen bestimmten Punkt enthält.

Metafile-Funktionen

Metafile-Funktionen schließen, kopieren, erzeugen, löschen und spielen Metafiles ab. Außerdem können sie Informationen über Metafiles liefern. Ein Metafile ist eine Sammlung von GDI-Befehlen, die gewünschten Text oder Bilder erzeugt.

Metafiles stellen eine bequeme Methode dar, Grafikbefehle zu speichern, die Text oder Bilder erzeugen. Metafiles sind besonders nützlich in Anwendungen, die einen bestimmten Text oder ein einzelnes Bild wiederholt verwenden. Sie sind auch geräteunabhängig; durch Erzeugen von Text oder Bildern mit GDI-Befehlen, die dann in einem Metafile abgelegt werden, kann eine Anwendung den Text oder die Bilder auf einer Vielzahl von Geräten neu erzeugen. Metafiles

sind außerdem in Anwendungen hilfreich, die Grafikinformationen an andere Anwendungen übergeben müssen. Die folgende Liste beschreibt kurz jede Metafile-Funktion:

CloseMetaFile	Schließt ein Metafile und erzeugt ein Metafile-Handle.
CopyMetaFile	Kopiert ein Quell-Metafile in eine Datei.
CreateMetaFile	Erzeugt einen Metafile-Bildschirmkontext.
DeleteMetaFile	Löscht ein Metafile aus dem Speicher.
EnumMetaFile	Listet die GDI-Aufrufe innerhalb eines Metafiles auf.
GetMetaFile	Erzeugt ein Handle zu einem Metafile.
GetMetaFileBits	Speichert ein Metafile als Sammlung von Bits in einem globalen Speicherblock.
PlayMetaFile	Spielt die Inhalte eines bestimmten Metafiles ab.
PlayMetaFileRecord	Spielt einen Metafile-Record ab.
SetMetaFileBits	Erzeugt ein Speicher-Metafile.

Rechteck-funktionen

Rechteckfunktionen ändern Informationen über Rechtecke im Client-Bereich eines Fensters oder liefern entsprechende Informationen. In Windows wird ein Rechteck über eine RECT-Datenstruktur definiert. Die Struktur enthält zwei Punkte: die obere linke und die untere rechte Ecke des Rechtecks. Die Seiten eines Rechtecks werden von diesen beiden Punkten aufgespannt und liegen parallel zur x- und y-Achse. Die folgende Liste beschreibt kurz jede Rechtecksfunktion:

CopyRect	Erstellt eine Kopie eines existierenden Rechtecks.
EqualRect	Bestimmt, ob zwei Rechtecke identisch sind.
InflateRect	Vergrößert oder verkleinert das angegebene Rechteck.
IntersectRect	Findet den Überschneidungsbereich von zwei Rechtecken.
OffsetRect	Verschiebt ein gegebenes Rechteck.
PtInRect	Zeigt an, ob ein bezeichneter Punkt innerhalb eines Rechtecks liegt.
SetRectEmpty	Macht ein Rechteck zu einem leeren Rechteck (alle Koordinaten werden auf Null gesetzt).

UnionRect	Speichert die Vereinigung von zwei Rechtecken (das kleinste Rechteck, das die beiden angegebenen enthält).

Region-funktionen

Regionfunktionen erzeugen und verändern Informationen über Regionen und liefern solche Informationen. Ein Region ist ein elliptischer oder vieleckiger Bereich innerhalb eines Fensters, der mit grafischer Ausgabe gefüllt werden kann. Eine Anwendung benutzt diese Funktionen in Verbindung mit den Clipping-Funktionen zum Erzeugen von Clipping-Regionen. Die folgende Liste beschreibt kurz jede Regionfunktion:

CombineRgn	Vereint zwei vorhandene Regionen zu einer neuen.
CreateEllipticRgn	Erzeugt eine elliptische Region.
CreateEllipticRgnIndirect	Erzeugt eine elliptische Region.
CreatePolygonRgn	Erzeugt eine geschlossene vieleckige Region.
CreatePolyPolygonRgn	Erzeugt eine Region, die aus einer Reihe von Vielecken besteht. Diese werden so ausgefüllt, als ob sie ein einziges Vieleck wären.
CreateRectRgn	Erzeugt eine rechteckige Region.
CreateRectRgnIndirect	Erzeugt eine rechteckige Region aus einer bestehenden Datenstruktur.
CreateRoundRectRgn	Erzeugt eine rechteckige Region mit abgerundeten Ecken.
EqualRgn	Ermittelt, ob zwei Regionen identisch sind.
FillRgn	Füllt die angegebene Region mit einem Pinselmuster aus.
FrameRgn	Zeichnet eine Umrandung für eine bestimmte Region.
GetRgnBox	Liefert die Koordinaten des eine Region einschließenden Rechtecks.
InvertRgn	Invertiert die Farben in einer Region.
OffsetRgn	Verschiebt die angegebene Region.
PaintRgn	Füllt die Region mit dem ausgewählten Pinselmuster aus.
PtInRegion	Ermittelt, ob ein Punkt innerhalb einer Region liegt.

RectInRegion	Ermittelt, ob ein Teil eines Rechtecks innerhalb einer Region liegt.
SetRectRgn	Erzeugt eine rechteckige Region.

Schrift-funktionen

Schriftfunktionen erzeugen und löschen Schriften, wählen sie aus und liefern Information über sie. Eine Schrift ist eine Untermenge eines bestimmten Schrifttyps. Ein Schrifttyp ist eine Menge von Zeichen, die dasselbe Grundmuster haben. Die folgende Liste beschreibt kurz jede Schriftfunktion:

AddFontResource	Fügt eine Schriftressource aus der angegebenen Datei zur Systemschriftentabelle hinzu.
CreateFont	Erzeugt eine virtuelle Schrift, die die angegebenen Merkmale hat.
CreateFontIndirect	Erzeugt eine virtuelle Schrift, die die angegebenen Merkmale hat.
EnumFonts	Listet die auf einem vorgegebenen Gerät verfügbaren Schriften auf.
GetCharWidth	Liefert die Breite bestimmter Zeichen.
RemoveFontResource	Löscht eine Schriftressource aus der Schriftentabelle.
SetMapperFlags	Ändert den Algorithmus, den der Schriftenabbilder benutzt.

Text-funktionen

Textfunktionen liefern Textinformationen, ändern Textanordnung, und Textausrichtung und schreiben Text auf ein Gerät oder eine Bildschirmoberfläche. Die GDI verwendet die aktuelle Schrift zur Textausgabe. Die folgende Liste beschreibt kurz jede Textfunktion:

ExtTextOut	Schreibt eine Zeichenkette unter Verwendung der aktuellen Schrift in eine rechteckige Region. Die rechteckige Region kann undurchsichtig (mit der aktuellen Hintergrundfarbe ausgefüllt) und eine Clipping-Region sein.
GetTabbedTextExtent	Berechnet die Breite und Höhe einer Textzeile, die Tabulatorzeichen enthält.
GetTextAlign	Gibt eine Maske der Textanordnungsflags zurück.

GetTextExtent	Benutzt die aktuelle Schrift, um Breite und Höhe von Text zu berechnen.
GetTextFace	Kopiert den aktuellen Schriftnamen in einen Puffer.
GetTextMetrics	Füllt den Puffer mit Maßeinheiten für die ausgewählte Schrift.
SetTextAlign	Positioniert einen Textstring auf dem Bildschirm oder einem Gerät.
SetTextJustification	Richtet eine Textzeile aus.
TabbedTextOut	Schreibt eine Zeichenkette mit erweiterten Tabulatoren in der aktuellen Schrift.
TextOut	Schreibt eine Zeichenkette in der aktuellen Schrift.

Umgebungsfunktionen

Umgebungsfunktionen ändern Informationen über die einem Ausgabegerät zugeordnete Umgebung und liefern solche Informationen. Die folgende Liste beschreibt kurz die beiden Umgebungsfunktionen:

GetEnvironment	Kopiert Umgebungsinformationen in einen Puffer.
SetEnvironment	Kopiert Daten in die Umgebung, die einem angeschlossenen Gerät zugeordnet ist.

Zeichnungsattributfunktionen

Zeichnungsattributfunktionen betreffen das Aussehen der Windows-Ausgabe, die in vier Formen stattfinden kann: Linie, Pinsel, Bitmap und Text. Die folgende Liste beschreibt kurz jede Zeichnungsattributfunktion:

GetBkColor	Gibt die aktuelle Hintergrundfarbe zurück.
GetBkMode	Gibt den aktuellen Hintergrundmodus zurück.
GetPolyFillMode	Liefert den aktuellen Polygonfüllmodus.
GetROP2	Liefert den aktuellen Zeichenmodus.
GetStretchBltMode	Liefert den aktuellen Streckmodus.
GetTextColor	Liefert die aktuelle Textfarbe.
SetBkColor	Setzt die Hintergrundfarbe.

SetBkMode	Setzt den Hintergrundmodus.
SetPolyFillMode	Setzt den Polygonfüllmodus.
SetROP2	Setzt den aktuellen Zeichenmodus.
SetStretchBltMode	Setzt den Streckmodus.
SetTextColor	Setzt die Textfarbe.

Zeichenwerkzeug-funktionen

Zeichenwerkzeugfunktionen erzeugen und löschen die Zeichenwerkzeuge, die von der GDI für Ausgaben auf einem Gerät oder einer Bildschirmoberfläche verwendet werden. Die folgende Liste beschreibt kurz jede Zeichenwerkzeugfunktion:

CreateBrushIndirect	Erzeugt einen virtuellen Pinsel aus einem bestehenden.
CreateDIBPatternBrush	Erzeugt einen virtuellen Pinsel, der ein durch ein geräteunabhängiges Bitmap (DIB) definiertes Muster hat.
CreateHatchBrush	Erzeugt einen virtuellen Pinsel, der ein schraffiertes Muster hat.
CreatePatternBrush	Erzeugt einen virtuellen Pinsel, der ein durch ein Speicher-Bitmap definiertes Muster hat.
CreatePen	Erzeugt einen virtuellen Stift.
CreatePenIndirect	Erzeugt einen virtuellen Stift aus einer bestehenden Datenstruktur.
CreateSolidBrush	Erzeugt einen virtuellen Pinsel.
DeleteObject	Löscht einen virtuellen Stift, Pinsel, eine virtuelle Schrift, einen Bereich oder ein virtuelles Bitmap.
EnumObjects	Listet die verfügbaren Stifte oder Pinsel auf.
GetBrushOrg	Liefert den aktuellen Ursprung eines Pinsels für einen Gerätekontext.
GetObject	Kopiert die Bytes virtueller Daten, die ein Objekt definieren.
GetStockObject	Liefert ein Handle zu einem der vordefinierten verfügbaren Stifte, Pinsel, Schriften oder Farbpaletten.
SelectObject	Wählt ein Objekt als das aktuelle Objekt aus.

SetBrushOrg	Setzt den Ursprung aller ausgewählten Pinsel im angegebenen Gerätekontext.
UnrealizeObject	Weist die GDI an, den Ursprung des angegebenen Pinsels zurückzusetzen.

Linienausgabe-funktionen

Linienausgabefunktionen erzeugen einfache und komplexe Liniendarstellungen mit dem ausgewählten Stift. Die folgende Liste beschreibt kurz jede Linienausgabefunktion:

Arc	Zeichnet einen Bogen.
FrameRect	Zeichnet die vier Kanten eines Rechtecks mit dem angegebenen Pinsel (anders als die übrigen hier aufgeführten Funktionen).
LineDDA	Berechnet aufeinanderfolgende Punkte auf einer Linie.
LineTo	Zeichnet eine Linie mit dem ausgewählten Stift.
MoveTo	Verschiebt die aktuelle Position zu dem angegebenen Punkt.
Polyline	Zeichnet eine Anzahl von Liniensegmenten.

Funktionen der Windows-API

In die Zuständigkeit der nachfolgend aufgeführten Funktionen (die aus KERNEL.EXE stammen) fallen Speicherverwaltung, die Verwaltung von Ressourcen, die Steuerung des Ablaufs von Windows-Anwendungen und die Interaktion mit DOS.

Anwendungsausführungs-funktionen

Anwendungsausführungs-Tasks erlauben es einer Anwendung, ein anderes Programm auszuführen. Die folgende Liste beschreibt diese Funktionen kurz:

LoadModule	Führt eine Windows-Anwendung aus.
WinExec	Führt eine Windows- oder Nicht-Windows-Anwendung aus.

WinHelp	Läßt die Windows-Hilfeanwendung laufen und übergibt kontext- oder themenbezogene Informationen an Help.

Die Funktion *WinExec* bietet eine High-Level-Methode, jede Windows- oder DOS-Anwendung auszuführen. Die aufrufende Anwendung übergibt einen String, der den Namen der auszuführenden Datei, sämtliche Befehlsparameter und den Anfangsstatus des Anwendungsfensters enthält.

Die Funktion *LoadModule* arbeitet ähnlich, bietet aber mehr Kontrolle über das Environment, in der die Anwendung ausgeführt wird. Die aufrufende Anwendung übergibt den Namen der ausführbaren Datei und einen Parameterblock für die DOS-Funktion 4BH, Code 00H.

Die Funktion *WinHelp* führt das Windows-Hilfeprogramm aus und übergibt ihm optional Daten, um die Art der von der Anwendung angeforderten Hilfe festzulegen. Diese Daten bestehen entweder in einem Integerwert, der einen Kontextbezeichner in der Hilfedatei bestimmt oder in einem String, der ein Schlüsselwort der Hilfedatei enthält.

Atomverwaltungsfunktionen

Atomverwaltungsfunktionen erzeugen und manipulieren Atome. Atome sind Integerwerte, die Zeichenketten eindeutig bezeichnen. Sie sind hilfreich in Anwendungen, die viele Zeichenketten benutzen und in Anwendungen, die Speicher sparen müssen. Windows speichert Atome in Atomtabellen. Eine lokale Atomtabelle wird im Datensegment einer Anwendung angelegt; auf sie kann von anderen Anwendungen nicht zugegriffen werden. Die globale Atomtabelle kann gemeinsam benutzt werden und sie ist sinnvoll in Anwendungen, die dynamischen Datenaustausch (DDE) vornehmen. Die folgende Liste beschreibt kurz jede Atomverwaltungsfunktion:

AddAtom	Erzeugt ein Atom für eine Zeichenkette.
DeleteAtom	Erniedrigt eine Atomreferenz und löscht das Atom, wenn der Referenzzähler Null ist.
FindAtom	Liefert ein Atom, das einer Zeichenkette zugeordnet ist.
GetAtomHandle	Liefert ein Handle (bezogen auf den lokalen Heap) des Strings, der zu einem angegebenen Atom gehört.
GetAtomName	Kopiert die Zeichenkette, die einem Atom zugeordnet ist.
GlobalAddAtom	Erzeugt ein globales Atom für eine Zeichenkette.
GlobalDeleteAtom	Erniedrigt eine Atomreferenz und löscht das globale Atom, wenn der Referenzzähler Null ist.

GlobalFindAtom	Liefert ein globales Atom, das einer Zeichenkette zugeordnet ist.
GlobalGetAtomName	Kopiert die Zeichenkette, die einem globalen Atom zugeordnet ist.
InitAtomTable	Initialisiert eine Atom-Hash-Tabelle.
MAKEINTATOM	Makro, das einen Integer zur Verwendung als Funktionsargument liefert.

Betriebssystem-Interruptfunktionen

Betriebssystem-Interruptfunktionen erlauben es Assembleranwendungen, bestimmte DOS- und NETBIOS-Interrupts auszuführen, ohne den Interrupt direkt zu codieren. Dies stellt die Kompatibilität mit künftigen Microsoft-Produkten sicher. Die folgende Liste beschreibt diese Funktionen kurz:

DOS3Call	Löst einen DOS-Interrupt 21H aus (Funktionsanforderung).
NetBIOSCall	Löst einen NETBIOS-Interrupt 5CH aus.

Dateiein-/ausgabe-funktionen

Dateiein-/ausgabefunktionen erstellen, öffnen, lesen aus und schreiben in Dateien. Nachfolgend eine kurze Beschreibung der zur Verfügung stehenden Funktionen:

GetDriveType	Bestimmt, ob es sich um Disketten-, Festplatten- oder Netzlaufwerk handelt.
GetSystemDirectory	Liefert den Pfadnamen des Windows-Systemverzeichnisses (z.B. C:\WINDOWS\SYSTEM).
GetTempDrive	Gibt den Kennbuchstaben des Laufwerks zurück, das zur Speicherung temporärer Dateien am besten geeignet ist.
GetTempFileName	Erzeugt den Namen einer neuen temporären Datei.
GetWindowsDirectory	Liefert den Pfadnamen des Windows-Verzeichnisses (z.B. C:\WINDOWS).
_lclose	Schließt eine Datei.
_lcreat	Erzeugt eine neue Datei oder öffnet eine vorhandene Datei und kürzt sie auf Nullänge.
_llseek	Positioniert den Dateizeiger.

_lopen	Öffnet eine vorhandene Datei.
_lread	Liest Daten aus einer Datei.
_lwrite	Schreibt Daten in eine Datei.
OpenFile	Erzeugt, öffnet, löscht eine angegebene Datei oder öffnet sie erneut.
SetHandleCount	Verändert die Anzahl von Datei-Handles, die für eine Task verfügbar sind.

Debugging-Funktionen

Debugging-Funktionen helfen beim Auffinden von Programmierfehlern in Anwendungen und Bibliotheken. Nachstehend eine kurze Beschreibung dieser Funktionen:

DebugBreak	Erzwingt eine Unterbrechung im Debugger.
FatalAppExit	Zeigt »Nicht behebbarer Fehler im Anwendungsprogramm« im Meldungsfenster und beendet die Anwendung.
FatalExit	Stellt den aktuellen Status von Windows dar (wenn der Debugger aktiv ist) und erwartet Anweisungen zur weiteren Bearbeitung.
OutputDebugString	Übergibt eine Debugging-Botschaft an den Debugger oder an den Hilfsausgang (AUX), wenn der Debugger nicht aktiv ist.
ValidateCodeSegments	Ermittelt, ob Codesegmente durch zufällige Speicherüberschreibungen verändert wurden.
ValidateFreeSpaces	Überprüft freie Speichersegmente auf gültige Inhalte.

Dienstmakros und -funktionen

Dienstmakros und -funktionen geben Inhalte von Worten und Bytes zurück und erzeugen Unsigned long-Integerzahlen und Datenstrukturen. Die folgende Liste beschreibt kurz jedes Dienstmakro bzw. jede Dienstfunktion:

HIBYTE	Gibt das höherwertige Byte einer Integerzahl zurück.
HIWORD	Gibt das höherwertige WORD einer long-Integerzahl zurück.

LOBYTE	Gibt das niederwertige Byte einer Integerzahl zurück.
LOWORD	Gibt das niederwertige WORD einer long-Integerzahl zurück.
MAKEINTATOM	Liefert eine Integerzahl zur Verwendung als Funktionsargument.
MAKEINTRESOURCE	Konvertiert einen Integerwert in einen long-Zeiger auf einen String, wobei das höherwertige WORD des long-Zeigers auf Null gesetzt ist.
MAKELONG	Erzeugt einen long-Integer.
MAKEPOINT	Konvertiert einen long-Wert, der die x- und y-Koordinaten eines Punktes in einer Datenstruktur POINT enthält.
MulDiv	Multipliziert zwei Wortlängenwerte und dividiert das Ergebnis durch einen dritten. Das Ergebnis wird - gerundet auf die nächste Integerzahl - zurückgegeben.
PALETTEINDEX	Konvertiert eine Integerzahl in einen Palettenindexwert *COLORREF*.
PALETTERGB	Konvertiert drei Werte für Rot, Grün und Blau in einen palettenbezogenen Wert *RGB_COLORREF*.
RGB	Konvertiert drei Werte für Rot, Grün undBlau in einen expliziten Wert *RGB_COLORREF*.

Initialisierungsdatei-funktionen

Initialisierungsdateifunktionen holen Informationen aus und kopieren Informationen in die Windows-Initialisierungsdatei WIN.INI. Die Bearbeitung privater Initialisierungsdateien ist ebenfalls möglich. Eine Windows-Initialisierungsdatei ist eine spezielle ASCII-Datei, die Wertepaare von Schlüsselanweisungen enthält, die Laufzeitoptionen für Anwendungen darstellen. Die folgende Liste beschreibt jede Initialisierungsdateifunktion:

GetPrivateProfileInt	Gibt einen Integer aus einem Abschnitt einer privaten Initialisierungsdatei zurück.
GetPrivateProfileString	Gibt eine Zeichenkette aus einem Abschnitt einer privaten Initialisierungsdatei zurück.
GetProfileInt	Gibt einen Integerwert aus einem Abschnitt der Datei WIN.INI zurück.
GetProfileString	Gibt eine Zeichenkette aus einem Abschnitt der Datei WIN.INI zurück.

WritePrivateProfileString	Kopiert oder löscht eine oder mehrere Zeilen in privaten Initialisierungsdateien.
WriteProfileString	Kopiert oder löscht eine oder mehrere Zeilen in WIN.INI.

Eine Anwendung sollte eine private (anwendungsspezifische) Initialisierungsdatei benutzen, um Informationen aufzuzeichnen, die nur diese Anwendung betreffen. Dies verbessert die Geschwindigkeit der Anwendung und von Windows selbst, da die Menge der Information verkleinert wird, die Windows lesen muß, wenn es auf die Initialisierungsdatei zugreift. Eine Anwendung sollte nur dann Informationen in WIN.INI hinterlegen, wenn sie die Windows-Umgebung oder andere Anwendungen betreffen; in solchen Fällen sollte die Anwendung die Botschaft *WM_WININICHANGE* an alle Fenster höchster Ebene senden.

Klang-funktionen

Klangfunktionen erzeugen Töne und Musik für den Klanggenerator des Systems. Die folgende Liste beschreibt kurz jede Tonfunktion:

CloseSound	Schließt das Klangausgabegerät nach dem Entleeren der Klangschlangen und gibt die Puffer frei.
CountVoiceNotes	Gibt die Anzahl der Noten in der bezeichneten Schlange zurück.
GetThresholdEvent	Gibt einen long-Zeiger auf das Schwellenflag zurück.
GetThresholdStatus	Gibt den Schwellenereignisstatus für jeden Klang zurück.
OpenSound	Öffnet das Klangausgabegerät zum exklusiven Gebrauch.
SetSoundNoise	Setzt die Quelle und die Dauer eines Tons im Klangausgabegerät.
SetVoiceAccent	Plaziert einen Ton in der Klangschlange.
SetVoiceEnvelope	Plaziert die Klanghüllkurve in der Klangschlange.
SetVoiceNote	Plaziert eine Note in der angegebenen Klangschlange.
SetVoiceQueueSize	Reserviert eine angegebene Zahl von Bytes für die Klangschlange.
SetVoiceSound	Plaziert die angegebene Tonfrequenz und -dauer in einer Klangschlange.

SetVoiceThreshold	Setzt den Schwellenwert für einen angegebenen Klang.
StartSound	Startet das Abspielen jeder Klangschlange.
StopSound	Beendet das Abspielen aller Klangschlangen und leert ihre Inhalte.
SyncAllVoices	Plaziert eine Synchronisationsmarke in jeder Klangschlange.
WaitSoundState	Wartet, bis das Klangausgabegerät den angegebenen Status erreicht hat.

Modulverwaltungsfunktionen

Modulverwaltungsfunktionen verändern Windows-Module und liefern Informationen über sie. Windows-Module sind ladbare, ausführbare Einheiten von Programmcode und Daten. Die folgende Liste beschreibt kurz jede Modulverwaltungsfunktion:

FreeLibrary	Verringert den Referenzzähler einer Bibliothek um Eins und löscht sie aus dem Speicher, wenn der Referenzzähler Null ist.
FreeModule	Verringert den Referenzzähler eines Moduls um Eins und löscht es aus dem Speicher, wenn der Referenzzähler Null ist.
FreeProcInstance	Löscht den Eintrag einer Funktionsinstanz an einer Adresse.
GetCodeHandle	Bestimmt, welches Codesegment eine bestimmte Funktion enthält.
GetInstanceData	Kopiert Daten vom Offset einer Instanz zum Offset einer anderen Instanz.
GetModuleFileName	Kopiert einen Moduldateinamen.
GetModuleHandle	Gibt das Modul-Handle eines Moduls zurück.
GetModuleUsage	Gibt den Referenzzähler eines Moduls zurück.
GetProcAddress	Gibt die Adresse einer Funktion in einem Modul zurück.
GetVersion	Gibt die aktuelle Windows-Versionsnummer zurück.
LoadLibrary	Lädt ein Bibliotheksmodul.
MakeProcInstance	Gibt die Adresse einer Funktionsinstanz zurück.

Ressourcenverwaltungs-funktionen

Ressourcenverwaltungsfunktionen finden und laden Anwendungsressourcen aus einer unter Windows ausführbaren Datei. Eine Ressource kann ein Zeiger, Symbol, Bitmap, String oder eine Schrift sein. Die folgende Liste beschreibt kurz jede Ressourcenverwaltungsfunktion:

AccessResource	Öffnet die angegebene Ressource.
AllocResource	Reserviert nicht initialisierten Speicher für eine Ressource.
FindResource	Bestimmt die Position einer Ressource.
FreeResource	Löscht eine geladene Ressource aus dem Speicher.
LoadAccelerators	Lädt eine Accelerator-Tabelle.
LoadBitmap	Lädt eine Bitmap-Ressource.
LoadCursor	Lädt eine Zeigerressource.
LoadIcon	Lädt eine Symbolressource.
LoadMenu	Lädt eine Menüressource.
LoadResource	Lädt eine Ressource.
LoadString	Lädt eine String-Ressource.
LockResource	Liefert die absolute Speicheradresse einer Ressource.
SetResourceHandler	Richtet eine Funktion zum Laden von Ressourcen ein.
SizeofResource	Übergibt die Größe einer Ressource in Bytes.
UnlockResource	Entsperrt eine Ressource.

Schnittstellen-funktionen

Schnittstellenfunktionen führen Übertragungen über die seriellen und parallelen I/0-Anschlüsse aus. Die folgende Liste beschreibt kurz jede Übertragungsfunktion:

BuildCommDCB	Füllt einen Gerätesteuerblock mit Steuerbefehlen.
ClearCommBreak	Löscht den Status der Übertragungsunterbrechung eines Schnittstellengeräts.
CloseComm	Schließt eine Schnittstelle nach dem Senden des aktuellen Puffers.

EscapeCommFunction	Weist ein Gerät an, eine erweiterte Funktion auszuführen.
FlushComm	Gibt alle Zeichen aus einem Puffer an eine Schnittstelle aus.
GetCommError	Füllt einen Puffer mit dem Schnittstellenstatus.
GetCommEventMask	Liefert eine Ereignismaske und löscht sie dann.
GetCommState	Füllt einen Puffer mit einem Gerätesteuerblock.
OpenComm	Öffnet eine Schnittstelle.
ReadComm	Liest die Bytes aus einer Schnittstelle in einen Puffer.
SetCommBreak	Setzt einen Unterbrechungsstatus am Schnittstellengerät.
SetCommEventMask	Liefert eine Ereignismaske und setzt sie dann am Schnittstellengerät.
SetCommState	Versetzt eine Schnittstelle in den Status, den der Gerätesteuerblock vorgibt.
TransmitCommChar	Plaziert ein Zeichen am Kopf der Sendeschlange.
UngetCommChar	Bestimmt, welches Zeichen als nächstes gelesen werden soll.
WriteComm	Schreibt die Bytes aus einem Puffer an ein Schnittstellengerät.

Segment-funktionen

Segmentfunktionen reservieren, konvertieren und geben Selektoren frei. Sie sperren und entsperren Speicherblöcke, auf die sich Selektoren beziehen. Die folgende Liste beschreibt kurz jede Selektorfunktion:

AllocDStoCSAlias	Nimmt einen Datensegmentselektor entgegen und gibt einen Codesegmentselektor zurück, der benutzt werden kann, um Code in einem Datensegment auszuführen.
AllocSelector	Reserviert einen neuen Selektor.
ChangeSelector	Erzeugt einen temporären Codeselektor, der einem gegebenen Datenselektor entspricht, oder einen temporären Datenselektor, der einem gegebenen Codeselektor entspricht.
DefineHandleTable	Erzeugt eine private Handle-Tabelle, die Windows automatisch aktualisiert.

FreeSelector	Gibt einen Selektor frei, der ursprünglich durch *AllocSelector* oder *AllocDStoCSAlias* belegt wurde.
GetCodeInfo	Liefert Informationen über ein Codesegment.
GlobalFix	Verhindert, daß ein globaler Speicherblock im linearen Speicher verschoben wird.
GlobalPageLock	Sperrt den Speicher seitenweise, der dem angegebenen globalen Selektor zugeordnet ist und erhöht seinen Seitensperrenzähler. Speicher, der seitenweise gesperrt ist, kann nicht verschoben oder seitenweise auf Diskette ausgelagert werden.
GlobalPageUnlock	Verringert den Seitensperrenzähler eines Speicherblocks. Wenn der Seitensperrenzähler Null erreicht, dann kann der Speicherbereich verschoben und seitenweise auf Diskette ausgelagert werden.
GlobalUnfix	Entsperrt einen globalen Speicherblock, der vorher mit der Funktion *GlobalFix* gesperrt wurde.
LockSegment	Sperrt ein Segment im Speicher.
UnlockSegment	Entsperrt ein Segment, das vorher mit der Funktion *LockSegment* gesperrt wurde.

Anmerkung: Eine Anwendung sollte diese Funktionen nicht benutzen, wenn es nicht absolut notwendig ist. Der Einsatz dieser Funktionen verletzt alle gebräulichen Windows-Programmierpraktiken.

Speicherverwaltungsfunktionen

Speicherverwaltungsfunktionen verwalten den Systemspeicher. Es gibt zwei Kategorien von Funktionen: jene, die globalen Speicher verwalten und die, die lokalen Speicher verwalten. Globaler Speicher ist der gesamte Speicherplatz im System, der nicht von einer Anwendung belegt wurde oder vom System reserviert ist. Lokaler Speicher ist der Speicher innerhalb des Datensegments einer Windows-Anwendung. Die folgende Liste beschreibt kurz jede Speicherverwaltungsfunktion:

DefineHandleTable	Erzeugt eine private Handle-Tabelle im Standarddatensegment einer Anwendung.
GetFreeSpace	Liefert die Anzahl von Bytes, die auf dem globalen Heap verfügbar sind.
GetWinFlags	Liefert Information über die Systemspeicherkonfiguration.

GlobalAlloc	Weist Speicher im globalen Heap zu.
GlobalCompact	Verdichtet globalen Speicher, um freie Bytes zu erzeugen.
GlobalDiscard	Verwirft einen globalen Speicherblock, wenn der Sperrenzähler Null ist. Das Handle des Speicherblocks wird nicht ungültig gemacht.
GlobalDosAlloc	Reserviert globalen Speicher, auf den DOS zugreifen kann, wenn das Gerät im Real oder Protected Mode läuft.
GlobalDosFree	Gibt globalen Speicher frei, der vorher von GlobalDosAlloc reserviert wurde.
GlobalFlags	Gibt die Flags und den Sperrenzähler eines globalen Speicherblocks zurück.
GlobalFree	Löscht einen globalen Speicherblock und macht dessen Handle ungültig.
GlobalHandle	Liefert das Handle eines globalen Speicherobjekts.
GlobalLock	Liefert den Zeiger auf einen globalen Speicherblock, der durch ein Handle festgelegt ist. Außer bei Objekten im Standard- oder erweiterten 386-Modus, denen nicht das Attribut »discardable« zugeordnet ist (d.h. deren Speicherbereich nicht verworfen werden kann), wird der Block im Speicher an der vorgegebenen Adresse gesperrt und der Sperrenzähler um Eins erhöht.
GlobalLRUNewest	Verschiebt ein globales Speicherobjekt an die neueste, zuletzt verwendete (LRU) Position.
GlobalLRUOldest	Verschiebt ein globales Speicherobjekt an die älteste, zuletzt verwendete (LRU) Position.
GlobalNotify	Installiert eine Benachrichtigungsprozedur für die aktuelle Task.
GlobalReAlloc	Weist einen globalen Speicherblock neu zu.
GlobalSize	Gibt die Größe eines globalen Speicherblocks in Bytes zurück.
GlobalUnlock	Macht den durch *GlobalLock* gelieferten Zeiger auf einen Speicherblock ungültig. Ist dem Block das Attribut »discardable« zugeordnet oder wird im Real Mode gearbeitet, wird der Sperrenzähler um Eins verringert.

GlobalUnwire	Verringert den von der Funktion *GlobalWire* gesetzten Sperrenzähler und entsperrt den Speicherblock, wenn der Zähler Null ist.
GlobalWire	Verschiebt ein Objekt in den unteren Speicherbereich und erhöht den Sperrenzähler.
LimitEmsPages	Begrenzt die Größe des *Expanded Memory*, den Windows einer Anwendung zuweist.
LocalAlloc	Reserviert Speicher im lokalen Heap.
LocalCompact	Verdichtet lokalen Speicher.
LocalDiscard	Verwirft einen lokalen Speicherblock, wenn der Sperrenzähler Null ist. Das Handle des Speicherblocks bleibt gültig.
LocalFlags	Gibt den Speichertyp eines lokalen Speicherblocks zurück.
LocalFree	Löscht einen lokalen Speicherblock aus dem Speicher, wenn der Sperrenzähler Null ist und macht dessen Handle ungültig.
LocalHandle	Liefert das Handle eines lokalen Speicherobjekts.
LocalInit	Initialisiert einen lokalen Heap im angegebenen Segment.
LocalLock	Sperrt einen lokalen Speicherblock, indem sie seinen Sperrenzähler erhöht.
LocalReAlloc	Weist einen lokalen Speicherblock neu zu.
LocalShrink	Verkleinert den lokalen Heap.
LocalSize	Gibt die Größe eines lokalen Speicherblocks in Bytes zurück.
LocalUnlock	Entsperrt einen lokalen Speicherblock.
LockData	Sperrt das aktuelle Datensegment im Speicher.
LockSegment	Sperrt ein angegebenenes Datensegment im Speicher.
SetSwapAreaSize	Vergrößert den Speicher, den eine Anwendung für ihr Codesegment reserviert.
SwitchStackBack	Gibt den Stack der aktuellen Task an deren Datensegment zurück, nachdem es vorher von der Funktion *SwitchTasksBack* umadressiert wurde.
SwitchStackTo	Ändert den Stack der aktuellen Task in das angegebene Datensegment. Dabei kann es sich z.B.

	um das Datensegment einer dynamischen Linkbibliothek (DLL) handeln.
UnlockData	Entsperrt das aktuelle Datensement.
UnlockSegment	Entsperrt ein bestimmtes Datensegment.

Stringmanipulationsfunktionen

Stringmanipulationsfunktionen übersetzen Strings aus einem Zeichensatz in einen anderen, bestimmen und konvertieren die Schreibweise eines Strings, bestimmen, ob ein Zeichen alphabetisch oder alphanumerisch ist, finden benachbarte Zeichen in einem String und führen andere Stringmanipulationen aus.

Die folgende Liste beschreibt kurz jede Stringmanipulationsfunktion:

AnsiLower	Konvertiert einen String in Kleinschreibung.
AnsiLowerBuff	Konvertiert eine Zeichenkette in einem Puffer in Kleinschreibung.
AnsiNext	Gibt einen long-Zeiger auf das nächste Zeichen in einem String zurück.
AnsiPrev	Gibt einen long-Zeiger auf das vorherige Zeichen in einem String zurück.
AnsiToOem	Konvertiert einen ANSI-String in eine OEM-Zeichenkette.
AnsiToOemBuff	Konvertiert einen ANSI-String in einem Puffer in eine OEM-Zeichenkette.
AnsiUpper	Konvertiert eine Zeichenkette in Großschreibung.
AnsiUpperBuff	Konvertiert eine Zeichenkette in einem Puffer in Großschreibung.
IsCharAlpha	Bestimmt, ob ein Zeichen alphabetisch ist.
IsCharAlphaNumeric	Bestimmt, ob ein Zeichen alphanumerisch ist.
IsCharLower	Bestimmt, ob ein Zeichen in Kleinschreibung vorliegt.
IsCharUpper	Bestimmt, ob ein Zeichen in Großschreibung vorliegt.
lstrcat	Verkettet zwei Strings, die durch long-Zeiger bezeichnet werden.
lstrcmp	Führt einen Vergleich zweier Strings unter Berücksichtigung der Klein- und Großschrei-

	bung durch. Die Strings werden durch long-Zeiger bezeichnet.
lstrcmpi	Führt einen Vergleich zweier Strings ohne Berücksichtigung der Klein- und Großschreibung durch. Die Strings werden durch long-Zeiger bezeichnet.
lstrcpy	Kopiert einen String auf einen anderen. Beide Strings werden durch long-Zeiger bezeichnet.
lstrlen	Bestimmt die Länge eines Strings, der durch einen long-Zeiger bezeichnet wird.
OemToAnsi	Konvertiert eine OEM-Zeichenkette in einen ANSI-String.
OemToAnsiBuff	Konvertiert eine OEM-Zeichenkette in einem Puffer in einen ANSI-String.
ToAscii	Übersetzt einen virtuellen Tastencode in das oder die entsprechenden ANSI-Zeichen.
wsprintf	Formatiert und speichert eine Folge von Zeichen und Werten in einem Puffer. Formatargumente werden separat übergeben.
wvsprintf	Formatiert und speichert eine Folge von Zeichen und Werten in einem Puffer. Formatargumente werden über ein Array von Zeigern übergeben.

Task-Funktionen

Task-Funktionen verändern den Ausführungsstatus von Tasks, geben Informationen, die einer Task zugeordnet sind, zurück und liefern Informationen über die Umgebung, in der die Task läuft. Eine Task ist ein einzelner Windows-Anwendungsaufruf. Die folgende Liste beschreibt kurz jede Task-Funktion:

Catch	Kopiert die aktuelle Ausführungsumgebung in einen Puffer.
ExitWindows	Leitet die Standardschließprozedur für Windows ein.
GetCurrentPDB	Gibt die aktuelle DOS-Programmdatenbasis (PDB) zurück, auch Programmsegmentpräfix (PSP) genannt.
GetCurrentTask	Gibt das Task-Handle der aktuellen Task zurück.
GetDOSEnvironment	Liefert den Environment-String der aktuell laufenden Task.

GetNumTasks	Gibt die Anzahl von Tasks zurück, die gerade im System laufen.
SetErrorMode	Legt fest, ob Windows die gemeldeten Fehler der DOS-Funktion 24H bearbeitet oder es der Anwendung überläßt, sie zu bearbeiten.
Throw	Stellt die Ausführungsumgebung mit den angegebenen Werten wieder her.
Yield	Unterbricht die aktuelle Task und startet eine wartende Task.

Schnittstellenfunktionen der Fenster-Verwaltung (WMI)

Diese Funktionen sind in USER.EXE zusammengefaßt. Neben der Verwaltung der Fenster umfaßt ihr Aufgabenbereich noch die Kommunikation mit dem Anwender.

Anzeige- und Verschiebefunktionen

Anzeige- und Verschiebefunktionen zeigen, verbergen und verschieben Fenster und ermitteln Informationen über die Anzahl und die Position von Fenstern auf dem Bildschirm. Die folgende Liste beschreibt kurz jede Anzeige- und Verschiebungsfunktion:

ArrangeIconicWindows	Ordnet verkleinerte untergeordnete Fenster (Symbole) an.
BeginDeferWindowPos	Initialisiert Speicher, der von der Funktion *DeferWindowPos* benutzt wird.
BringWindowToTop	Bringt ein Fenster an die oberste Position in einem Stapel überlappender Fenster.
CloseWindow	Verbirgt das angegebene Fenster oder verkleinert es.
DeferWindowPos	Speichert Positionierungsinformation für ein Fenster, das von der Funktion *EndDeferWindowPos* verschoben oder dessen Größe verändert wird.
EndDeferWindowPos	Positioniert mehrere Fenster oder legt ihre Größe aufgrund von Informationen neu fest, die mit der Funktion *DeferWindowPos* gespeichert wurden.

GetClientRect	Kopiert die Koordinaten des Client-Bereichs eines Fensters.
GetWindowRect	Kopiert die Abmessungen eines Fensters.
GetWindowText	Kopiert eine Fensterüberschrift in einen Puffer.
GetWindowTextLength	Gibt die Länge (in Zeichen) der Überschrift oder des Texts des angegebenen Fensters zurück.
IsIconic	Gibt an, ob ein Fenster geöffnet oder geschlossen (Symbol) ist.
IsWindowVisible	Bestimmt, ob das angegebene Fenster sichtbar ist.
IsZoomed	Bestimmt, ob ein Fenster vergrößert ist.
MoveWindow	Ändert Größe und Position eines Fensters.
OpenIcon	Öffnet das bezeichnete Fenster.
SetWindowPos	Ändert die Größe, Position und Reihenfolge von untergeordneten oder Pop-up-Fenstern.
SetWindowText	Setzt den Text oder die Überschrift des Fensters.
ShowOwnedPopups	Zeigt alle Pop-up-Fenster oder blendet sie aus.
ShowWindow	Zeigt oder verschiebt das angegebene Fenster.

Bildlauf-funktionen

Bildlauffunktionen steuern den Bildlauf der Inhalte eines Fensters und die Bildlaufleisten des Fensters. Bildlauf ist das Verschieben von Daten in und aus dem Client-Bereich auf Veranlassung des Benutzers. Dies ist für den Benutzer eine Möglichkeit, ein Dokument oder eine Grafik in Teilen zu betrachten, wenn Windows nicht in der Lage ist, das gesamte Dokument oder die ganze Grafik in den Client-Bereich einzupassen. Eine Bildlaufleiste ermöglicht es dem Benutzer, den Bildlauf zu steuern. Die folgende Liste beschreibt kurz jede Bildlauffunktion:

GetScrollPos	Liefert die aktuelle Position des Bildlaufleistensymbols.
GetScrollRange	Kopiert die minimale und maximale Bildlaufleistenposition für die vorgegebenen Bildlaufleistenpositionen und den festgelegten Bildlauf.
ScrollDC	Verschiebt ein Rechteck aus Bits horizontal und vertikal.
ScrollWindow	Verschiebt die Inhalte des Client-Bereichs.
SetScrollPos	Setzt die Bildlaufleistenpositionsmarke.

SetScrollRange	Setzt die minimale und maximale Bildlaufleistenposition.
ShowScrollBar	Zeigt eine Bildlaufleiste und ihr Steuerfeld oder blendet sie aus.

Botschafts-funktionen

Botschaftsfunktionen lesen und verarbeiten Windows-Botschaften in einer Anwendungsschlange. Botschaften stellen eine Vielfalt von Eingaben an eine Windows-Anwendung dar. Eine Botschaft ist eine Datenstruktur, die einen Botschaftsbezeichner und Botschaftsparameter enthält. Der Inhalt der Parameter wechselt mit dem Botschaftstyp. Die folgende Liste beschreibt kurz jede Funktion:

CallWindowProc	Übermittelt Botschaftsinformationen an die angegebene Funktion.
DispatchMessage	Übermittelt eine Botschaft an eine Fensterfunktion des angegebenen Fensters.
GetMessage	Ruft eine Botschaft aus dem angegebenen Bereich von Botschaften ab.
GetMessagePos	Gibt die Position der Maus zu dem Zeitpunkt, zu dem die letzte Botschaft abgerufen wurde, zurück.
GetMessageTime	Gibt die Zeit zurück, zu der die letzte Botschaft abgerufen wurde.
InSendMessage	Bestimmt, ob die aktuelle Fensterfunktion eine Botschaft verarbeitet, die an sie mit einem Aufruf der Funktion *SendMessage* übergeben wurde.
PeekMessage	Untersucht die Anwendungsschlange und plaziert die Botschaft geeignet.
PostAppMessage	Stellt eine Botschaft in die Anwendungsschlange.
PostMessage	Stellt eine Botschaft in die Anwendungsschlange.
PostQuitMessage	Übermittelt eine Botschaft WM_QUIT an die Anwendung.
ReplyMessage	Antwortet auf eine Botschaft.
SendMessage	Sendet eine Botschaft an ein oder mehrere Fenster.
SetMessageQueue	Erzeugt eine neue Botschaftsschlange unterschiedlicher Größe.

TranslateAccelerator	Bearbeitet Tastatur-Accelerator für Menübefehle.
TranslateMDISysAccel	Verarbeitet Befehls-Accelerator für untergeordnete Fenster mit Multidokumentschnittstelle (MDI).
TranslateMessage	Übersetzt virtuelle Tastenfolgebotschaften in Zeichenbotschaften.
WaitMessage	Übergibt die Kontrolle an andere Anwendungen.
WinMain	Dient als Einsprungspunkt zur Ausführung einer Windows-Anwendung.

Caret-Funktionen

Caret-Funktionen betreffen das Windows-Caret, das aus einer blinkenden Linie, einem Block oder einem Bitmap besteht, womit die Position im Client-Bereich eines Fensters markiert wird. Das Caret ist besonders in textverarbeitenden Anwendungen zur Markierung einer Position im Text für Tastatureingaben nützlich. Diese Funktionen erstellen und zerstören das Caret, blenden es aus und ein und ändern die Blinkrate des Carets. Die folgende Liste beschreibt kurz jede Caret-Funktion:

CreateCaret	Erzeugt ein Caret.
DestroyCaret	Zerstört das aktuelle Caret.
GetCaretBlinkTime	Gibt die Blinkrate des Carets zurück.
GetCaretPos	Gibt die aktuelle Caret-Position zurück.
HideCaret	Löscht ein Caret aus einem gegebenen Fenster.
SetCaretBlinkTime	Stellt die Blinkrate des Carets ein.
SetCaretPos	Bewegt das Caret zur angegebenen Position.
ShowCaret	Zeigt das neu erzeugte Caret an oder stellt ein ausgeblendetes Caret wieder dar.

Dialogfenster-funktionen

Dialogfensterfunktionen erzeugen, verändern, testen und zerstören Dialogfenster und Steuerelemente innerhalb von Dialogfenstern. Ein Dialogfenster ist ein temporäres Fenster, das Windows für Eingaben zu besonderen Zwecken erzeugt und dann sofort nach Gebrauch wieder zerstört. Eine Anwendung benutzt ein Dialogfenster üblicherweise dazu, vom Benutzer zusätzliche Informationen über die aktuelle Befehlsauswahl anzufordern. Die folgende Liste beschreibt kurz die einzelnen Dialogfunktionen:

CheckDlgButton	Setzt/löscht eine Auswahlmarke oder ändert den Status eines dreistufigen Schalters.
CheckRadioButton	Wählt ein angegebenes Schaltfeld und löscht Auswahlmarkierungen von allen anderen.
CreateDialog	Erzeugt ein nicht-modales Dialogfenster.
CreateDialogIndirect	Erzeugt ein nicht-modales Dialogfenster aus einer Schablone.
CreateDialogIndirectParam	Erzeugt ein nicht-modales Dialogfenster aus einer Schablone und übergibt Daten, wenn es erstellt wird.
CreateDialogParam	Erstellt ein nicht-modales Dialogfenster und übergibt Daten, wenn es erstellt wird.
DefDlgProc	Ermöglicht Standardverarbeitung für alle Windows-Botschaften, die ein Dialogfenster einer privaten Fensterklasse nicht verarbeitet.
DialogBox	Erzeugt ein modales Dialogfenster.
DialogBoxIndirect	Erzeugt ein modales Dialogfenster aus einer Schablone.
DialogBoxIndirectParam	Erzeugt ein modales Dialogfenster aus einer Schablone und übergibt Daten, wenn es erzeugt wird.
DialogBoxParam	Erzeugt ein modales Dialogfenster und übergibt Daten, wenn es erzeugt wird.
DlgDirList	Füllt eine Liste mit den Dateinamen, die einem Pfadnamen mit Wildcards entsprechen.
DlgDirListComboBox	Füllt ein Kombinationsfenster mit Dateinamen, die einem Pfad entsprechen.
DlgDirSelect	Kopiert die aktuelle Auswahl aus einer Liste in einen String.
DlgDirSelectComboBox	Kopiert die aktuelle Auswahl aus einem Kombinationsfenster in einen String.
EndDialog	Gibt Ressourcen frei und zerstört Fenster, die einem modalen Dialogfenster zugeordnet sind.
GetDialogBaseUnits	Liefert die Basisdialogeinheiten, die von Windows beim Erzeugen eines Dialogfensters benutzt werden.
GetDlgCtrlID	Gibt den ID-Wert eines Steuerungsfensters zurück.

GetDlgItem	Liefert das Handle eines Eintrags im angegebenen Dialogfenster.
GetDlgItemInt	Übersetzt den Text eines Steuerelements in einen Integerwert.
GetDlgItemText	Kopiert den Text eines Steuerelements in einen String.
GetNextDlgGroupItem	Gibt das Fenster-Handle des nächsten Eintrags in einer Gruppe zurück.
GetNextDlgTabItem	Gibt das Fensterhandle des nächsten oder vorherigen Eintrags zurück.
IsDialogMessage	Bestimmt, ob eine Funktion für das angegebene Dialogfenster bestimmt ist.
IsDlgButtonChecked	Testet, ob ein Schalter markiert ist.
MapDialogRect	Konvertiert die Dialogfensterkoordinaten in Client-Koordinaten.
SendDlgItemMessage	Sendet eine Botschaft an ein Element innerhalb eines Dialogfensters.
SetDlgItemInt	Macht die Überschrift oder den Text eines Elements zu einem String, der einen Integerwert darstellt.
SetDlgItemText	Macht die Überschrift oder den Text eines Elements zu einem String.

Eigenschafts-funktionen

Eigenschaftsfunktionen erzeugen die Eigenschaftsliste eines Fensters und greifen auf diese zu. Eine Eigenschaftsliste ist ein Speicherbereich, der Handles für Daten enthält, die die Anwendung einem Fenster zuzuordnen wünscht. Die folgenden Liste beschreibt kurz jede Eigenschaftsfunktion:

EnumProps	Übergibt die Eigenschaften eines Fensters an eine Aufzählungsfunktion.
GetProp	Liefert ein Handle, das einem String zugeordnet ist, aus der Eigenschaftsliste.
RemoveProp	Löscht einen String aus der Eigenschaftsliste.
SetProp	Kopiert einen String und ein Daten-Handle in die Eigenschaftsliste eines Fensters.

Eingabe-funktionen

Eingabefunktionen sperren die Eingabe von Systemgeräten (Maus, Tastatur und Timer), übernehmen deren Steuerung oder definieren spezielle Aktionen, die Windows ausführt, wenn eine Anwendung Eingaben von einem Systemgerät erhält. Die folgende Liste beschreibt kurz jede Eingabefunktion:

EnableWindow	Schaltet Maus- und Tastatureingabe für die Anwendung ein und aus.
GetActiveWindow	Gibt ein Handle zum aktiven Fenster zurück.
GetCapture	Gibt ein Handle zum Fenster mit dem Mausfang zurück.
GetCurrentTime	Liefert die aktuelle Windows-Zeit.
GetDoubleClickTime	Liefert das aktuelle Doppelklickzeitintervall für die Maus.
GetFocus	Liefert das Handle des Fensters, das gerade den Fokus besitzt.
GetTickCount	Gibt die Anzahl der Timer-Ticks zurück, die seit dem Booten des Systems gezählt wurden.
IsWindowEnabled	Bestimmt, ob das Fenster Maus- und Tastatureingaben entgegennehmen kann.
KillTimer	Bricht das angegebene Timer-Ereignis ab.
ReleaseCapture	Hebt Mauseingabe auf und stellt die normale Eingabeverarbeitung wieder her.
SetActiveWindow	Macht ein Fenster zum aktiven Fenster.
SetCapture	Veranlaßt, daß die Mauseingabe ausschließlich an das angegebene Fenster gesendet wird.
SetDoubleClickTime	Setzt das Doppelklickzeitintervall für die Maus.
SetFocus	Weist dem angegebenen Fenster den Fokuszu.
SetSysModalWindow	Macht das angegebene Fenster zu einem systemmodalen Fenster.
SetTimer	Erzeugt ein Systemtimer-Ereignis.
SwapMouseButton	Vertauscht die Bedeutung der linken und der rechten Maustaste.

Fehler-funktionen

Fehlerfunktionen zeigen Fehler an und fordern den Anwender zu einer Reaktion auf. Die folgende Liste beschreibt kurz jede Fehlerfunktion:

FlashWindow	Läßt das Fenster aufblinken, indem es seinen aktiven bzw. inaktiven Zustand in das jeweilige Gegenteil verkehrt.
MessageBeep	Erzeugt einen Warnton am Systemlautsprecher.
MessageBox	Erzeugt ein Fenster mit der angegebenen Überschrift.

Fenstererstellungs-funktionen

Fenstererstellungsfunktionen dienen zum Erstellen, Löschen, Verändern und Informationsabruf von Fenstern. Die folgende Liste beschreibt kurz jede Fenstererstellungsfunktion:

AdjustWindowRect	Berechnet die Größe eines Fensters, um einen gegebenen Client-Bereich anzupassen.
AdjustWindowRectEx	Berechnet die Größe eines Fensters mit erweitertem Modus, um einen gegebenen Client-Bereich anzupassen.
CreateWindow	Erzeugt überlappte, untergeordnete und Pop-up-Fenster.
CreateWindowEx	Erzeugt überlappte, untergeordnete und Pop-up-Fenster mit erweitertem Stil.
DefDlgProc	Ermöglicht Standardverarbeitung für diejenigen Botschaften, die eine Anwendung nicht verarbeitet.
DefFrameProc	Ermöglicht Standardverarbeitung für diejenigen MDI-Rahmenfensterbotschaften, die eine Anwendung nicht verarbeitet.
DefMDIChildProc	Ermöglicht Standardverarbeitung für diejenigen MDI-Botschaften an untergeordnete Fenster, die eine Anwendung nicht verarbeitet.
DefWindowProc	Ermöglicht Standardverarbeitung für diejenigen Fensterbotschaften, die eine Anwendung nicht verarbeitet.
DestroyWindow	Zerstört ein Fenster.
GetClassInfo	Liefert Information über das angegebene Fenster.

GetClassLong	Liefert Fensterklasseninformation aus einer Struktur *WNDCLASS*.
GetClassName	Liefert eine Fensterklassenbezeichnung.
GetClassWord	Liefert Fensterklasseninformation aus einer Struktur *WNDCLASS*.
GetLastActivePopup	Bestimmt, welches Pop-up-Fenster, das von einem anderen Fenster besessen wird, zuletzt aktiv war.
GetWindowLong	Liefert Information über ein Fenster.
GetWindowWord	Liefert Information über ein Fenster.
RegisterClass	Registriert eine Fensterklasse.
SetClassLong	Ersetzt eine long-Integer-Information in einer Struktur *WNDCLASS*.
SetClassWord	Ersetzt eine WORD-Information in einer Struktur *WNDCLASS*.
SetWindowLong	Ändert ein long-Integer-Fensterattribut.
SetWindowWord	Ändert ein WORD-Fensterattribut.
UnregisterClass	Löscht eine Fensterklasse aus der Fensterklassentabelle.

Filterverwaltungsfunktionen

Filterverwaltungsfunktionen verwalten Systemfilter, die gemeinsam genutzte Ressourcen sind, die einen bestimmten Typ von Filterfunktion installieren. Eine Filterfunktion ist eine von der Anwendung bereitgestellte Callback-Funktion, die von der Funktion *SetWindowsHook* bestimmt wird. Sie verarbeitet Ereignisse, bevor sie die Botschaftsschleife einer Anwendung erreichen. Windows sendet Botschaften, die von einem bestimmten Ereignistyp erzeugt wurden, an Filterfunktionen, die mit demselben Filtertyp installiert sind. Die folgende Liste beschreibt kurz jede Filterverwaltungsfunktion:

CallMsgFilter	Übergibt eine Botschaft und andere Daten an die aktuelle Botschaftsfilterfunktion.
DefHookProc	Ruft die nächste Filterfunktion in einer Filterfunktionskette auf.
SetWindowsHook	Installiert eine System- und/oder Anwendungsfilterfunktion.
UnhookWindowsHook	Entfernt eine Windows-Filterfunktion aus einer Kette von Filterfunktionen.

Hardware-funktionen

Hardwarefunktionen ändern den Status von Eingabegeräten und liefern Statusinformation. Windows benutzt die Maus und die Tastatur als Eingabegeräte. Die folgende Liste beschreibt kurz jede Hardwarefunktion:

EnableHardwareInput	Schaltet Maus- und Tastatureingabe für die gesamte Anwendung ein und aus.
GetAsyncKeyState	Gibt Information über den Tastenzustand auf Interrupt-Ebene zurück.
GetInputState	Gibt TRUE zurück, wenn eine Maus- oder Tastatureingabe vorliegt.
GetKBCodePage	Bestimmt, welche OEM/ANSI-Tabellen geladen sind.
GetKeyboardState	Kopiert ein Array, das den Status der Tastaturtasten enthält.
GetKeyNameText	Liefert einen String, der die Bezeichnung einer Taste aus der Liste enthält, die der Tastaturtreiber verwaltet.
GetKeyState	Liefert den Status einer virtuellen Taste.
MapVirtualKey	Nimmt einen virtuellen Tastencode oder Scancode für eine Taste entgegen und gibt den zugehörigen Scancode, virtuellen Tastencode oder ASCII-Wert zurück.
OemKeyScan	Bildet die OEM-ASCII-Codes 0 bis 0x0FF auf OEM-Scancodes und Shift-Status ab.
SetKeyboardState	Setzt den Status von Tastaturtasten durch Ändern von Werten in einem Array.
VkKeyScan	Übersetzt ein ANSI-Zeichen in den zugehörigen virtuellen Tastencode und Shift-Status für die aktuelle Tastatur.

Informations-funktionen

Informationsfunktionen holen Information über Anzahl und Position von Fenstern auf dem Bildschirm. Die folgende Liste beschreibt kurz jede Informationsfunktion:

AnyPopup	Zeigt an, ob ein Pop-up-Fenster existiert.
ChildWindowFromPoint	Stellt fest, welches untergeordnete Fenster einen bestimmten Punkt enthält.

EnumChildWindows	Zählt die untergeordneten Fenster auf, die zu einem angegebenen übergeordneten Fenster gehören.
EnumTaskWindows	Zählt alle Fenster, die einer bestimmten Task zugeordnet sind.
EnumWindows	Zählt alle Fenster auf dem Bildschirm.
FindWindow	Gibt das Handle eines Fensters mit der gegebenen Klasse und Überschrift zurück.
GetNextWindow	Gibt ein Handle zum nächsten oder vorherigen Fenster zurück.
GetParent	Liefert das Handle des angegebenen übergeordneten Fensters.
GetTopWindow	Gibt das Handle eines Hauptfensters zurück.
GetWindow	Gibt ein Handle aus der Liste der Fenste-Verwaltung zurück.
GetWindowTask	Gibt das Handle einer Task zurück, die einem Fenster zugeordnet ist.
IsChild	Gibt an, ob ein Fenster der Abkömmling eines vorgegebenen Fensters ist.
IsWindow	Stellt fest, ob ein Fenster ein gültiges existierendes Fenster ist.
SetParent	Ändert das übergeordnete Fenster eines untergeordneten Fensters.
WindowFromPoint	Bezeichnet das Fenster, daß einen angegebenen Punkt enthält.

Mauszeiger-funktionen

Zeigerfunktionen setzen, verschieben, zeigen und begrenzen den Zeiger und blenden ihn gegebenenfalls aus. Der Zeiger ist ein Bitmap, das auf dem Bildschirm dargestellt wird und eine aktuelle Position anzeigt. Die folgende Liste beschreibt kurz jede Zeigerfunktion:

ClipCursor	Beschränkt den Zeiger auf ein vorgegebenes Rechteck.
CreateCursor	Erzeugt einen Zeiger aus zwei Bitmasken.
DestroyCursor	Zerstört einen Zeiger, der von der Funktion *CreateCursor* erzeugt wurde.
GetCursorPos	Speichert die Zeigerposition in Bildschirmkoordinaten.

LoadCursor	Lädt einen Zeiger aus der Ressourcendatei.
SetCursor	Setzt die Zeigerform.
SetCursorPos	Setzt die Position des Zeigers.
ShowCursor	Verringert oder erhöht den Zähler der Zeigerdarstellung.

Menüfunktionen

Menüfunktionen erzeugen, verändern und zerstören Menüs. Ein Menü ist ein Eingabehilfsmittel in einer Windows-Anwendung, das Benutzern eine oder mehrere Auswahlmöglichkeiten bietet, die sie mit der Maus oder der Tastatur auswählen können. Eine Option in einer Menüleiste kann ein Pop-up-Menü darstellen und jede Option in einem Pop-up-Menü kann ein weiteres Pop-up-Menü anzeigen. Außerdem kann ein Pop-up-Menü irgendwo auf dem Bildschirm erscheinen. Die folgende Liste beschreibt kurz jede Menüfunktion:

AppendMenu	Fügt eine Menüoption einem Menü hinzu.
CheckMenuItem	Setzt oder löscht Häkchen neben Pop-up-Menüoptionen.
CreateMenu	Erzeugt ein leeres Menü.
CreatePopupMenu	Erzeugt ein leeres Pop-up-Menü.
DeleteMenu	Löscht eine Menüoption und zerstört alle zugeordneten Pop-up-Menüs.
DestroyMenu	Zerstört das bezeichnete Menü.
DrawMenuBar	Zeichnet eine Menüleiste neu.
EnableMenuItem	Schaltet eine Menüoption ein und aus oder versetzt sie in den nicht aktivierbaren Zustand.
GetMenu	Liefert ein Handle zum Menü eines angegebenen Fensters.
GetMenuCheck-MarkDimensions	Liefert die Abmessungen des Bitmaps der Standardmenüauswahlmarke.
GetMenuItemCount	Gibt die Anzahl von Optionen in einem Menü zurück.
GetMenuItemID	Gibt den Bezeichner der Option zurück.
GetMenuState	Liefert den Status einer Menüoption.
GetMenuString	Kopiert ein Menü-Label in einen String.
GetSubMenu	Liefert das Menü-Handle eines Pop-up-Menüs.

GetSystemMenu	Greift zum Kopieren und Verändern auf das Systemmenü zu.
HiliteMenuItem	Hebt eine Menüoption in einer Menüleiste hervor oder nimmt die Hervorhebung zurück.
InsertMenu	Fügt eine Menüoption in ein Menü ein.
LoadMenuIndirect	Lädt eine Menüressource.
ModifyMenu	Verändert eine Menüoption.
RemoveMenu	Löscht eine Option aus einem Menü, zerstört sie aber nicht.
SetMenu	Bestimmt ein neues Menü für ein Fenster.
SetMenuItemBitmaps	Ordnet Bitmaps einer Menüoption zu, um anzuzeigen, ob sie ausgewählt ist.
TrackPopupMenu	Stellt ein Pop-up-Menü an einer angegebenen Position auf dem Bildschirm dar und verfolgt den Dialogeingriff des Benutzers.

Systemfunktionen

Systemfunktionen geben Informationen über das Maßsystem des Systems, die Farbe und die Zeit. Die folgende Liste beschreibt kurz jede Systemfunktion:

GetCurrentTime	Gibt die Zeit zurück, die seit dem Booten des Systems vergangen ist.
GetSysColor	Liefert die Systemfarbe.
GetSystemMetrics	Liefert Information über das Maßsystem des Systems.
SetSysColors	Ändert eine oder mehrere Systemfarben.

Zeichenfunktionen

Zeichenfunktionen bereiten ein Fenster zum Zeichnen vor und führen einige nützliche Allzweckgrafikfunktionen aus. Obwohl alle Zeichenfunktionen speziell für den Systembildschirm vorgesehen sind, können einige für andere Ausgabegeräte benutzt werden. Die folgende Liste beschreibt kurz jede Zeichenfunktion:

BeginPaint	Bereitet ein Fenster zum Zeichnen vor.
DrawFocusRect	Zeichnet ein Rechteck in dem Stil, der den Focus anzeigt.

DrawIcon	Zeichnet ein Symbol.
DrawText	Zeichnet Zeichen eines angegebenen Strings.
EndPaint	Beendet das Zeichnen in ein Fenster.
ExcludeUpdateRgn	Verhindert das Zeichnen innerhalb ungültiger Bereiche eines Fensters.
FillRect	Füllt ein angegebenes Rechteck mit dem angegebenen Pinsel aus.
FrameRect	Zeichnet eine Umrandung für das angegebene Rechteck.
GetDC	Liefert den Bildschirmkontext für den Client-Bereich.
GetUpdateRect	Kopiert die Abmessungen des Rechtecks um einen Fensterbereich.
GetUpdateRgn	Kopiert den Aktualisierungsbereich eines Fensters.
GetWindowDC	Liefert den Bildschirmkontext für ein ganzes Fenster.
GrayString	Zeichnet die Zeichen eines Strings mit grauem Text.
InvalidateRect	Markiert ein Rechteck zur Neudarstellung.
InvalidateRgn	Markiert einen Bereich zur Neudarstellung.
InvertRect	Invertiert die Bildschirmbits des angegebenen Rechtecks.
ReleaseDC	Hebt einen Bildschirmkontext auf.
UpdateWindow	Benachrichtigt die Anwendung, wenn Teile eines Fensters neu gezeichnet werden müssen.
ValidateRect	Nimmt das bezeichnete Rechteck von der Neudarstellung aus.
ValidateRgn	Nimmt die bezeichnete Region von der Neudarstellung aus.

Zwischenablage-funktionen

Zwischenablagefunktionen übernehmen den Datenaustausch zwischen Windows-Anwendungsprogrammen. Die Zwischenablage ist der Ort für diesen Austausch; sie bietet einen Platz, von dem aus Anwendungen Daten-Handle an andere Anwendungen übergeben können. Die folgende Liste beschreibt kurz jede Zwischenablagefunktion:

ChangeClipboardChain	Löscht ein Fenster aus der Kette der Clipboard-Viewer.
CloseClipboard	Schließt die Zwischenablage.
EmptyClipboard	Leert die Zwischenablage und weist ihr einen neuen Eigner zu.
EnumClipboardFormats	Listet die verfügbaren Zwischenablageformate auf.
GetClipboardData	Liefert Daten aus der Zwischenablage.
GetClipboardFormatName	Liefert das Zwischenablageformat.
GetClipboardOwner	Liefert das Fenster-Handle, das dem aktuellen Zwischenablagebesitzer zugeordnet ist.
GetClipboardViewer	Liefert das Handle des ersten Fensters in der Kette der Clipboard-Viewer.
GetPriority-ClipboardFormat	Liefert Daten im ersten Format in einer vorrangigen Formatliste aus der Zwischenablage.
IsClipboardFormat-Available	Gibt TRUE zurück, wenn Daten im angegebenen Format verfügbar sind.
OpenClipboard	Öffnet die Zwischenablage.
RegisterClipboardFormat	Registriert ein neues Zwischenablageformat.
SetClipboardData	Kopiert ein Handle für Daten.
SetClipboardViewer	Fügt ein Handle in die Kette der Clipboard-Viewer hinzu.

Programmabbruchcodes

Die Debugger-Version von Windows erzeugt immer dann diagnostische Meldungen, wenn sie einen Fehler erkennt, der einen Systemabsturz bewirken würde. Jede diagnostische Meldung hat eine eindeutige Nummer, die die Ursache der Meldung und des potentiellen Absturzes anzeigt. Die folgende Tabelle enthält die meisten diagnostischen Botschaften mit ihren Kennummern und erklärt die Bedeutung jeder Meldung.

Tabelle 1.1 »Diagnosemeldungen«

Nummer(Hex)	Beschreibung
0000	NULL-Objekt-Handle an eine GDI-Funktion übergeben.
0001	Nicht genügend Speicherplatz für die angeforderte Zuweisung.
0002	Fehler beim Neuzuweisen von Speicherplatz.
0003	Speicher kann nicht freigegeben werden.
0004	Speicher kann nicht gesperrt werden.

Nummer(Hex)	Beschreibung
0005	Speicher kann nicht entsperrt werden.
0006	Ungültiges Handle an GDI-Funktion übergeben.
0007	Fenster-Handle ungültig.
0008	Alle Bildschirmkontexte sind besetzt.
000A	Zwischenablage schon offen.
000B	Zwischengespeicherter Bildschirmkontext von zerstörtem Fenster nicht freigegeben.
000C	Tastaturtreiber falsch initialisiert.
000D	Maustreiber falsch initialisiert.
000E	Bildschirmmodul nicht gültig.
000F	Entsperrtes Datensegment sollte gesperrt werden.
0100	Ungültiger lokaler Heap.
0140	Lokaler Heap ist besetzt.
0180	Ungültiges lokales Handle.
01C0	Zählwertüberlauf von *LocalLock*.
01F0	Zählwertunterlauf von *LocalUnlock*.
0200	Globaler Speicherfehler.
0240	Probleme mit kritischen Abschnitten (critical section problems).
0280	Ungültiges globales Handle.
02C0	Zählwertüberlauf von *GlobalLock*.
02F0	Zählwertunterlauf von *GlobalUnlock*.
0300	Task-Verzeichnisfehler.
0301	Ungültige Task-ID.
0302	Ungültiger Systemausstiegsaufruf.
0303	Ungültige BP-Registerkette.
0400	Fehler im dynamischen Loader/Linker.
0401	Fehler beim Booten.
0402	Fehler beim Laden eines Moduls.
0403	Ungültige Ordnungsreferenz.
0404	Ungültige Einsprungspunktreferenz.
0405	Ungültige Startprozedur.
0406	Ungültiges Modul-Handle.
0407	Ungültiger Verschiebungs-Record.
0408	Fehler beim Speichern einer Vorwärtsreferenz.
0409	Fehler beim Lesen von Segmentinhalten.
0410	Fehler beim Lesen von Verschiebungs-Records.
0411	Diskette für angegebene Datei einlegen.
0412	Fehler beim Lesen einer nicht residenten Tabelle.
04FF	INT 3F-Handler kann Segment nicht laden.
0500	Fehler in der Ressourcenverwaltung oder der Benutzerlaufzeitanalyse.
0501	Fehlende Ressourcentabelle.
0502	Falsche Ressourcentyp.
0503	Falsche Ressourcenname.
0504	Falsche Ressourcendatei.
0505	Fehler beim Lesen einer Ressource.
0600	Atomverwaltungsfehler.
0700	Ein-/Ausgabefehler.

Referenz aller Windows-Funktionen

Für jede Funktion werden die im folgenden Beispiel dargestellten Informationen aufgeführt. Im einzelnen sind dies die Syntaxbeschreibung sowie eine Beschreibung der ausgeführten Operation, der dabei verwendeten Parameter und des Rückgabewerts.

Alle Funktionen verwenden die im Gegensatz zur C-Konvention effizientere Pascal-Aufrufsequenz. Sie müssen als *FAR PASCAL* deklariert werden.

Beispiel

Syntax	`RTYPE Beispiel(ATYPE AParam, BTYPE BParam);`
Beschreibung	Eine kurze Beschreibung der Verwendung der Funktion *Beispiel*. *AParam:* Beschreibung des ersten Parameters. *BParam:* Beschreibung des zweiten Parameters.
Rückgabewert	Beschreibt den Inhalt des Rückgabewerts *RTYPE*, der von *Beispiel* in Abhängigkeit von der erfolgreichen Ausführung der Funktion zurückgegeben wird.
Hinweis	Enthält Informationen darüber, was beim Aufruf dieser Funktion zu beachten ist.
Callback-Funktion	Gegebenenfalls eine Beschreibung der Callback-Funktion, die von der Funktion *Beispiel* benötigt bzw. bereitgestellt wird.
Querverweis	Ein Hinweis auf verwandte Themen, Funktionen, Botschaften, Typen und Konstanten.

AccessResource

Syntax	`int AccessResource(HANDLE Instance, HANDLE ResInfo);`
Beschreibung	Diese Funktion öffnet die angegebene Ressourcendatei und stellt den Dateizeiger den Anfang der gewünschten Ressource, um der Anwendung das Lesen dieser Ressource zu ermöglichen. *AccessResource* liefert ein DOS-Datei-Handle, das in nachfolgenden Dateiaufrufen zum Laden der Ressource verwendet werden kann. Die Datei wird nur zum Lesen geöffnet. Anwendungen, die Ressourcen mit Hilfe von *AccessResource* einlesen, müssen die Ressourcendatei anschließend durch Aufruf der Funktion *_lclose* wieder schließen. *Instance:* Bezeichnet die Instanz des Moduls, in dessen ausführbarer Datei die Ressource enthalten ist.

ResInfo: Bezeichnet die gewünschte Ressource. Dieses Handle sollte mit Hilfe der Funktion *FindResource* erzeugt werden.

Rückgabewert Bei erfolgreicher Ausführung wird das DOS-Datei-Handle zur vorgegebenen Ressourcendatei zurückgeliefert. Der Rückgabewert ist -1, wenn die Ressource nicht gefunden werden kann.

Hinweis *AccessResource* braucht die verfügbaren DOS-Datei-Handles möglicherweise völlig auf und verursacht Fehler, wenn die geöffnete Datei nach dem Zugriff auf die Ressource nicht wieder geschlossen wird.

Querverweis *FindResource*

AddAtom

Syntax
```
ATOM AddAtom(LPSTR Str);
```

Beschreibung Diese Funktion fügt den durch den Parameter *Str* bezeichneten nullterminierten String in die Atomtabelle ein. Gleichzeitig wird ein neues Atom erzeugt, das den String bezeichnet. Das Atom kann in nachfolgenden Aufrufen der Funktion *GetAtomName* dazu verwendet werden, den String aus der Atomtabelle abzurufen.

AddAtom speichert nur eine einzige Kopie des vorgegebenen Strings in der Atomtabelle. Existiert der String bereits in der Tabelle, gibt die Funktion das zugehörige Atom zurück und erhöht den Referenzzähler des Strings um 1.

Str: Zeigt auf den in die Tabelle einzufügenden nullterminierten String.

Rückgabewert Bei erfolgreicher Ausführung der Funktion wird das erzeugte Atom zurückgegeben, andernfalls ist der Rückgabewert -1.

Hinweis Die von *AddAtom* zurückgegebenen Atome nehmen Werte zwischen 0xC000 und 0xFFFF an. Bei Atomen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Querverweis *GetAtomName*

AddFontResource

Syntax
```
int AddFontResource(LPSTR FileName);
```

Beschreibung Diese Funktion fügt die Schriftressource aus der durch den Parameter *FileName* bezeichneten Datei in die Schriftentabelle von Windows ein. Die Schrift kann anschließend von jeder Anwendung verwendet werden.

FileName: Zeigt auf einen String, der den Namen der Schriftressourcendatei oder ein Handle zum geladenen Modul ent-

hält. Zeigt *FileName* auf den Namen der Schriftressourcendatei, muß es sich um einen nullterminierten String im DOS-Dateinamensformat handeln. Die Namenserweiterung muß ebenfalls angegeben sein. Enthält *FileName* ein Handle, so liegt es im niederwertigen WORD, das höherwertige WORD ist 0.

Rückgabewert Es wird die Anzahl der hinzugefügten Schriften zurückgeliefert. Wurden keine Schriften geladen, ist der Rückgabewert 0.

Hinweis Jede Anwendung, die Schriften in die Schriftentabelle von Windows einfügt oder aus ihr entfernt, sollte diese Änderung anderen Fenstern mitteilen. Hierzu wird die Funktion *SendMessage* mit auf -1 gesetztem Parameter *Wnd* verwendet, die die Botschaft *WM_FONTCHANGE* an alle Hauptfenster übergibt. Anwendungen sollten alle durch sie hinzugefügten Schriftressourcen entfernen, wenn diese nicht mehr benötigt werden.

Querverweis *WM_FONTCHANGE*

AdjustWindowRect

Syntax
```
void AdjustWindowRect(LPRECT Rect, LONG Style,
BOOL Menu);
```

Beschreibung Diese Funktion berechnet die Größe des Fensters auf Grundlage der gewünschten Größe des Client-Bereichs. Danach kann das Fensterrechteck an die Funktion *CreateWindow* übergeben werden, die ein Fenster mit einem Client-Bereich der gewünschten Größe erzeugt.

Ein Client-Rechteck ist das kleinstmögliche den Client-Bereich umschließende Rechteck. Ein Fensterrechteck ist das kleinstmögliche das Fenster vollständig umschließende Rechteck. Die Größe des berechneten Fensterrechtecks hängt vom Fensterstil ab sowie davon, ob das Fenster ein Menü besitzt.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die Koordinaten des Client-Rechtecks enthält.

Style: Bestimmt den Stil des Fensters, dessen Client-Rechteck konvertiert werden soll.

Menu: Ungleich Null (*TRUE*), wenn das Fenster ein Menü hat.

Rückgabewert Keiner.

Hinweis Diese Funktion geht von einer einzelnen Menüzeile aus. Wenn sich das Menü über mehrere Zeilen erstreckt, sind die Koordinaten nicht korrekt.

Querverweis *CreateWindow*

AdjustWindowRectEx

Syntax
```
void AdjustWindowRectEx(LPRECT Rect, LONG Style,
BOOL Menu, DWORD ExStyle);
```

Beschreibung Diese Funktion berechnet die benötigte Größe eines Fensterrechtecks des erweiterten Stils auf Grundlage der gewünschten Größe des Client-Rechtecks. Anschließend kann das Fensterrechteck an die Funktion *CreateWindowEx* übergeben werden, die ein Fenster mit einem Client-Bereich der gewünschten Größe erzeugt.

Ein Client-Rechteck ist das kleinstmögliche den Client-Bereich umschließende Rechteck. Ein Fensterrechteck ist das kleinstmögliche das Fenster vollständig umschließende Rechteck.

Die Größe des berechneten Fensterrechtecks hängt vom Fensterstil ab sowie auch davon, ob das Fenster ein Menü besitzt.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die Koordinaten des Client-Rechtecks enthält.

Style: Bestimmt den Stil des Fensters, dessen Client-Rechteck konvertiert werden soll.

Menu: Ungleich Null (*TRUE*), wenn das Fenster ein Menü hat.

ExStyle: Bestimmt den erweiterten Stil des zu erzeugenden Fensters.

Rückgabewert Keiner.

Hinweis Diese Funktion geht von einer einzelnen Menüzeile aus. Wenn sich das Menü über mehrere Zeilen erstreckt, sind die Koordinaten nicht korrekt.

Querverweis *CreateWindowEx*

AllocDStoCSAlias

Syntax
```
WORD AllocDStoCSAlias(WORD Selector);
```

Beschreibung Diese Funktion übernimmt den Selektor eines Datensegments und gibt den Selektor eines Codesegments zurück. Wird im Protected Mode gearbeitet, führt der Versuch, Code in einem Datensegment auszuführen, zu einem nicht behebbaren Fehler in der Anwendung. *AllocDStoCSAlias* erlaubt die Ausführung von Code, der von der Anwendung im eigenen Stack-Segment erzeugt wurde.

Die Anwendung muß den neuen Selektor durch Aufruf der Funktion *FreeSelector* freigeben.

Selector: Bestimmt den Selektor des Datensegments.

Rückgabewert Der Rückgabewert ist der dem Selektor des Datensegments entsprechende Selektor des Codesegments. Kann die Funktion keinen neuen Selektor zuweisen, ist der Rückgabewert 0.

Hinweis Windows überwacht Segmentverschiebungen nicht. Aus diesem Grund muß das Datensegment fest sein und darf keinesfalls verschoben werden. Eine Verschiebung des Datensegments könnte fehlerhafte Veränderungen des Codesegments nach sich ziehen.

Mit der Funktion *ChangeSelector* kann der einem Daten-Selektor entsprechende Code-Selektor ebenfalls ermittelt werden.

Anwendungen sollten *AllocDStoCSAlias* nur in Ausnahmefällen verwenden. Die Verwendung dieser Funktion verletzt alle empfohlenen Praktiken der Windows-Programmierung.

AllocResource

Syntax
```
HANDLE AllocResource(HANDLE Instance, HANDLE ResInfo,
DWORD Size);
```

Beschreibung Diese Funktion reserviert nichtinitialisierten Speicher für die übergebene Ressource. Diese Reservierung muß für alle Ressourcen vorgenommen werden. *AllocResource* wird vor dem Laden einer Ressource durch die Funktion *LoadResource* aufgerufen.

Instance: Bezeichnet die Instanz des Moduls, in dessen ausführbarer Datei die Ressource enthalten ist.

ResInfo: Bezeichnet die gewünschte Ressource. Dieses Handle sollte mit Hilfe der Funktion *FindResource* erzeugt werden.

Size: Bestimmt eine Anzahl Bytes, die für die Ressource reserviert werden. Die Reservierung unterbleibt, wenn 0 vorgegeben wird.

Rückgabewert Der Rückgabewert bezeichnet den Speicherblock, der für die Ressource reserviert wurde.

Querverweis *FindResource, LoadResource*

AllocSelector

Syntax
```
WORD AllocSelector(WORD Selector);
```

Beschreibung Diese Funktion erzeugt einen neuen Selektor. Enthält der Parameter *Selector* einen gültigen Selektor, gibt *AllocSelector* eine genaue Kopie dieses Selektors als neuen Selektor zurück. Wenn *Selector* NULL ist, wird ein neuer, nichtinitialisierter Selektor zurückgegeben.

Anwendungen müssen den neuen Selektor durch Aufruf der Funktion *FreeSelector* freigeben.

Selector: Bestimmt entweder den zu kopierenden Selektor oder ist NULL, wenn *AllocSelector* einen neuen, nichtinitialisierten Selektor zuweisen soll.

Rückgabewert Der Rückgabewert ist entweder die Kopie eines bereits existierenden Selektors oder ein neuer, nichtinitialisierter Selektor. Konnte die Funktion keinen neuen Selektor zuweisen, ist der Rückgabewert 0.

Hinweis Der Aufruf von *AllocSelector* ermöglicht es Anwendungen, den zugewiesenen Selektor an die Funktion *ChangeSelector* zu übergeben.

Anwendungen sollten diese Funktion nur in Ausnahmefällen verwenden. Ihre Verwendung verletzt alle empfohlenen Praktiken der Windows-Programmierung.

AnimatePalette

Syntax
```
void AnimatePalette(HPALETTE hPalette, WORD StartIndex,
WORD NumEntries, LPPALETTEENTRY Palette);
```

Beschreibung Diese Funktion ersetzt Einträge in der durch den Parameter *Palette* bezeichneten virtuellen Farbpalette. Ruft eine Anwendung die Funktion *AnimatePalette* auf, ist eine Aktualisierung des Client-Bereichs nicht nötig, da Windows die neuen Einträge unmittelbar in die Systemfarbpalette abbildet.

hPalette: Bezeichnet die virtuelle Farbpalette.

StartIndex: Bestimmt den ersten zu verändernden Eintrag in der Farbpalette.

NumEntries: Bestimmt die Anzahl der zu verändernden Einträge in der Farbpalette.

Palette: Zeigt auf das erste Element eines Arrays von *PALETTEENTRY*-Datenstrukturen, durch die die Einträge ersetzt werden sollen. Die Einträge werden durch *StartIndex* und *NumEntries* identifiziert.

Rückgabewert Keiner.

Hinweis *AnimatePalette* ändert Einträge nur dann, wenn im zugehörigen Feld *palPaletteEntry* der *LOGPALETTE*-Datenstruktur das Flag *PC_RESERVED* gesetzt ist. *LOGPALETTE* definiert die aktuelle virtuelle Farbpalette. Die Funktion *CreatePalette* erzeugt eine virtuelle Farbpalette.

Querverweis *CreatePalette*

AnsiLower

Syntax
```
LPSTR AnsiLower(LPSTR Str);
```

Beschreibung Diese Funktion konvertiert den vorgegebenen String in Kleinbuchstaben. Die Konvertierung wird durch den Sprachtreiber auf Grundlage der aktuell ausgewählten Sprache durchgeführt. Die Auswahl der Sprache wird vom Anwender im Setup oder in der Systemsteuerung vorgenommen.

Str: Zeigt auf einen nullterminierten String oder bestimmt ein einzelnes Zeichen. Bestimmt *Str* ein einzelnes Zeichen, liegt es im niederwertigen Byte des niederwertigen WORD, das höherwertige WORD ist 0.

Rückgabewert Wenn der Funktionsparameter ein String ist, zeigt der Rückgabewert auf den konvertierten String. Andernfalls ist der Rückgabewert ein 32-Bit-Wert, der das konvertierte Zeichen im niederwertigen Byte des niederwertigen WORD enthält.

AnsiLowerBuff

Syntax
```
WORD AnsiLowerBuff(LPSTR Str, WORD Length);
```

Beschreibung Diese Funktion konvertiert den String im Puffer in Kleinbuchstaben. Die Konvertierung wird durch den Sprachtreiber auf Grundlage der aktuell ausgewählten Sprache durchgeführt. Die Auswahl der Sprache wird vom Anwender im Setup oder in der Systemsteuerung vorgenommen.

Str: Zeigt auf einen Puffer, der ein oder mehr Zeichen enthält.

Length: Bestimmt die Anzahl der Zeichen im durch den Parameter *Str* bezeichneten Puffer. Ist *Length* 0, beträgt die Länge 64 KByte (65536).

Rückgabewert Der Rückgabewert bestimmt die Länge des konvertierten Strings.

AnsiNext

Syntax
```
LPSTR AnsiNext(LPSTR CurrentChar);
```

Beschreibung Diese Funktion positioniert auf das nächste Zeichen in einem String.

CurrentChar: Zeigt auf ein Zeichen in einem nullterminierten String.

Rückgabewert Der Rückgabewert zeigt auf das nächste Zeichen im String. Existiert kein weiteres Zeichen, zeigt der Rückgabewert auf das Schlußzeichen des Strings.

Hinweis *AnsiNext* wird für das Bewegen durch Strings verwendet, deren Zeichen aus mindestens zwei Bytes bestehen, also beispielsweise Strings, deren Zeichen dem japanischen Zeichensatz entnommen sind.

AnsiPrev

Syntax
```
LPSTR AnsiPrev(LPSTR Start, LPSTR CurrentChar);
```

Beschreibung Die Funktion positioniert auf das vorhergehende Zeichen in einem String.

Start: Zeigt auf den Anfang des Strings.

CurrentChar: Zeigt auf ein Zeichen in dem nullterminierten String.

Rückgabewert Der Rückgabewert zeigt auf das vorhergehende Zeichen in einem String. Ist der Parameter *CurrentChar* identisch mit dem Parameter *Start*, zeigt der Rückgabewert auf das erste Zeichen des Strings.

Hinweis *AnsiPrev* wird für das Bewegen durch Strings verwendet, deren Zeichen aus mindestens zwei Bytes bestehen, also beispielsweise Strings, deren Zeichen dem japanischen Zeichensatz entnommen sind.

AnsiToOem

Syntax
```
int AnsiToOem(LPSTR AnsiStr, LPSTR OemStr);
```

Beschreibung Diese Funktion übersetzt den durch *AnsiStr* bezeichneten String vom ANSI-Zeichensatz in den OEM-Zeichensatz. Der String kann länger als 64 KByte sein.

AnsiStr: Zeigt auf einen nullterminierten String aus dem ANSI-Zeichensatz.

OemStr: Zeigt auf das Ziel, in das der String kopiert wird. Sind *OemStr* und *AnsiStr* identisch, wird der String an Ort und Stelle übersetzt.

Rückgabewert Der Rückgabewert ist immer -1.

AnsiToOemBuff

Syntax
```
void AnsiToOemBuff(LPSTR AnsiStr, LPSTR OemStr,
int Length);
```

Beschreibung Diese Funktion übersetzt den String im durch *AnsiStr* bezeichneten Puffer vom ANSI-Zeichensatz in den OEM-Zeichensatz.

AnsiStr: Zeigt auf einen Puffer, der ein oder mehr Zeichen des ANSI-Zeichensatzes enthält.

OemStr: Zeigt auf das Ziel, in das der String kopiert wird. *AnsiStr* und *OemStr* dürfen auf denselben Puffer zeigen.

Length: Bestimmt die Anzahl der Zeichen im durch den Parameter *AnsiStr* bezeichneten Puffer. Ist *Length* 0, beträgt die Länge 64 KByte (65536).

Rückgabewert Keiner.

AnsiUpper

Syntax
```
LPSTR AnsiUpper(LPSTR Str);
```

Beschreibung Diese Funktion konvertiert den vorgegebenen String in Großbuchstaben. Die Konvertierung wird durch den Sprachtreiber auf Grundlage der aktuell ausgewählten Sprache durchgeführt. Die Auswahl der Sprache wird vom Anwender im Setup oder in der Systemsteuerung vorgenommen.

Str: Zeigt auf einen nullterminierten String oder bestimmt ein einzelnes Zeichen. Bestimmt *Str* ein einzelnes Zeichen, liegt es im niederwertigen Byte des niederwertigen WORD, das höherwertige WORD ist 0.

Rückgabewert Wenn der Funktionsparameter ein String ist, zeigt der Rückgabewert auf den konvertierten String. Andernfalls ist der Rückgabewert ein 32-Bit-Wert, der das konvertierte Zeichen im niederwertigen Byte des niederwertigen Wortes enthält.

AnsiUpperBuff

Syntax
```
WORD AnsiUpperBuff(LPSTR Str, WORD Length);
```

Beschreibung Diese Funktion konvertiert den String im Puffer in Großbuchstaben. Die Konvertierung wird durch den Sprachtreiber auf Grundlage der aktuell ausgewählten Sprache durchgeführt. Die Auswahl der Sprache wird vom Anwender im Setup oder in der Systemsteuerung vorgenommen.

Str: Zeigt auf einen Puffer, der ein oder mehr Zeichen enthält.

Length: Bestimmt die Anzahl der Zeichen im durch den Parameter *Str* bezeichneten Puffer. Ist *Length* 0, beträgt die Länge 64 KByte (65536).

Rückgabewert Der Rückgabewert gibt die Länge des konvertierten Strings an.

AnyPopup

Syntax
```
BOOL AnyPopup(void);
```

Beschreibung Diese Funktion bestimmt, ob ein Pop-up-Fenster auf dem Bildschirm existiert. Die Funktion sucht den gesamten Windows-Bildschirm ab, nicht nur den Client-Bereich des Aufrufers. *AnyPopup* gibt auch dann einen Wert ungleich 0 zurück, wenn das Pop-up-Fenster völlig durch ein anderes Fenster verdeckt wird. Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn ein Pop-up-Fenster existiert, andernfalls ist er Null.

AppendMenu

Syntax
```
BOOL AppendMenu(HMENU Menu, WORD Flags, WORD IDNewItem,
LPSTR NewItem);
```

Beschreibung Diese Funktion fügt einen neuen Eintrag am Ende eines Menüs an. Anwendungen können den Status des Eintrags durch die Werte im Parameter *Flags* festlegen.

Menu: Bezeichnet das zu verändernde Menü.

Flags: Enthält Informationen über den Status des hinzugefügten Menüeintrags. Er besteht aus einem oder mehreren Werten. Eine Liste der möglichen Werte findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »MF_XXXX«.

IDNewItem: Bestimmt entweder den Befehlsbezeichner des neuen Menüeintrags oder das Handle des Pop-up-Menüs, wenn *Flags* auf *MF_POPUP* gesetzt ist.

NewItem: Bestimmt den Inhalt des neuen Menüeintrags. Wird ein Bitmap als Eintrag verwendet, enthält *NewItem* das zugehörige Handle im niederwertigen WORD. Die Interpretation des Parameters *NewItem* hängt vom Inhalt des Parameters *Flags* ab:

Flags	NewItem
MF_BITMAP	Enthält das Handle des Bitmap *HBITMAP* im niederwertigen WORD.
MF_OWNERDRAW	Enthält einen 32-Bit-Wert mit zusätzlichen, dem Menüeintrag assoziierten Daten. Dieser ist für die Anwendung im Feld *itemData* der durch den Parameter *lParam* der Botschaften *WM_MEASUREITEM* und *WM_DRAWITEM* bezeichneten Struktur *MEASUREITEMSTRUCT* verfügbar.

Flags	NewItem
	Diese Botschaften werden übergeben, wenn der Menüeintrag erstmals dargestellt oder verändert wird.
MF_STRING	Enthält einen Zeiger auf einen nullterminierten String.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich ausgeführt werden konnte, andernfalls ist er Null.

Hinweis Wird ein Menü verändert (unabhängig davon, ob sich das Menü in einem dargestellten Fenster befindet oder nicht), sollte die Anwendung die Funktion *DrawMenuBar* aufrufen.

Jede der folgenden Gruppen zeigt Flags, die sich gegenseitig ausschließen und nicht gleichzeitig verwendet werden sollten:

MF_BYCOMMAND und *MF_BYPOSITION*,
MF_DISABLED, *MF_ENABLED* und *MF_GRAYED*,
MF_BITMAP, *MF_STRING* und *MF_OWNERDRAW*,
MF_MENUBARBREAK und *MF_MENUBREAK*,
MF_CHECKED und *MF_UNCHECKED*

Querverweis *DrawMenuBar, SetMenuItemBitmaps, WM_DRAWITEM, WM_MEASUREITEM*

Arc

Syntax
```
BOOL Arc(HDC DC, int X1, int Y1, int X2, int Y2,
int X3, int Y3, int X4, int Y4);
```

Beschreibung Diese Funktion zeichnet einen Ellipsenbogen der vollständigen Ellipse. Der Mittelpunkt der vollständigen Ellipse entspricht dem Mittelpunkt des umgebenden Rechtecks, das durch die Punkte (*X1, Y1*) und (*X2, Y2*) festgelegt wird. Die Schnittpunkte der durch (*X3, Y3*) nach (*X4, Y4*) definierten Linie sind Start- und Endpunkt des zu zeichnenden Bogens. Der Ellipsenbogen wird mit dem ausgewählten Stift im Uhrzeigersinn gezeichnet. Er wird nicht ausgemalt, da ein Bogen keinen umschlossenen Bereich definiert.

DC: Bezeichnet den Gerätekontext.

X1, Y1: Bestimmt die virtuelle x- bzw. y-Koordinate der oberen linken Ecke des umgebenden Rechtecks.

X2, Y2: Bestimmt die virtuelle x- bzw. y-Koordinate der unteren rechten Ecke des umgebenden Rechtecks.

X3, Y3: Bestimmt die virtuelle x- bzw. y-Koordinate des Startpunkts des Kreisbogens. Dieser Punkt muß nicht genau auf dem Kreisbogen liegen.

X4, Y4: Bestimmt die virtuelle x- bzw. y-Koordinate des Endpunkts des Kreisbogens. Dieser Punkt muß nicht genau auf dem Kreisbogen liegen.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der Kreisbogen gezeichnet wird, andernfalls Null.

Hinweis Die Breite des Rechtecks ist *X2 - X1* und darf 32767 Einheiten nicht übersteigen. Diese Begrenzung gilt ebenfalls auch für die Höhe (*Y2 - Y1*).

Querverweis *Chord*

ArrangeIconicWindows

Syntax
```
WORD ArrangeIconicWindows(HWND Wnd);
```

Beschreibung Diese Funktion ordnet die Symbole der Fenster an, die in dem durch den Parameter *Wnd* bezeichneten Fenster liegen.

Wnd: Bezeichnet das übergeordnete Fenster.

Rückgabewert Der Rückgabewert ist die Höhe einer Symbolzeile. Er ist 0, wenn keine Symbole gefunden wurden.

Hinweis Anwendungen, die eigene Symbole verwalten, rufen *ArrangeIconicWindows* auf, um sie in einem Client-Fenster anzuordnen. Mit ihr werden auch die Symbole auf dem Desktop neu angeordnet. Die Funktion *GetDesktopWindow* liefert das Fenster-Handle des Desktop-Fensters. Um verkleinerte untergeordnete MDI-Fenster in einem MDI-Client-Bereich anzuordnen, übergibt die Anwendung die Botschaft *WM_MDIICONARRANGE* an das MDI-Client-Fenster.

Querverweis *GetDesktopWindow*

BeginDeferWindowPos

Syntax
```
HANDLE BeginDeferWindowPos(int NumWindows);
```

Beschreibung Diese Funktion reserviert Speicher, der eine Datenstruktur für mehrere Fensterpositionen aufnehmen soll und gibt dann das Handle dieser Struktur zurück.

Die Funktion *DeferWindowPos* informiert die Datenstruktur über die Zielposition des zu verschiebenden Fensters. Die Funktion *EndDeferWindowPos* übernimmt die in der Struktur abgelegten Daten und positioniert die Fenster sofort neu.

NumWindows: Bestimmt die anfängliche Anzahl von Fenstern, deren Positionsdaten in der Datenstruktur abgelegt werden. Die Funktion *DeferWindowPos* vergrößert die Datenstruktur, wenn dies nötig ist.

Rückgabewert Der Rückgabewert bezeichnet die Datenstruktur für die Fensterpositionen. Er ist 0, wenn die Systemressourcen nicht ausreichen, um Speicher für die Struktur zu reservieren.

Querverweis *DeferWindowPos, EndDeferWindowPos*

BeginPaint

Syntax
```
HDC BeginPaint(HWND Wnd, LPPAINTSTRUCT PaintStruct);
```

Beschreibung Diese Funktion bereitet das angegebene Fenster zum Zeichnen vor. Sie füllt die durch den Parameter *PaintStruct* bezeichnete Struktur mit entsprechenden Informationen. Diese Struktur enthält das Handle des Gerätekontextes eines Fensters, eine *RECT*-Datenstruktur, die das kleinstmögliche, den zu aktualisierenden Bereich umschließende Rechteck enthält und ein Flag, das bestimmt, ob der Hintergrund gelöscht wird.

BeginPaint setzt die Clipping-Region des Gerätekontextes, um jeden Bereich außerhalb der zu aktualisierenden Region auszuschließen. Die zu aktualisierende Region wird durch die Funktionen *InvalidateRect* oder *InvalidateRgn* gesetzt.

Das System setzt diese Region nach jeder Größenveränderung, Verschiebung, Erzeugung oder nach anderen Operationen, die den Client-Bereich beeinflußen. Ist die zu aktualisierende Region zum Löschen markiert, sendet *BeginPaint* die Botschaft *WM_ERASEBKGND* an das Fenster.

Anwendungen sollten *BeginPaint* nur als Reaktion auf die Botschaft *WM_PAINT* aufrufen. Jedem Aufruf von *BeginPaint* muß ein Aufruf der Prozedur *EndPaint* folgen.

Wnd: Bezeichnet das neu zu zeichnende Fenster.

PaintStruct: Zeigt auf eine *PAINTSTRUCT*-Datenstruktur, die die Zeichnungsinformationen aufnehmen soll (also den Gerätekontext, das zu aktualisierende Rechteck, usw.).

Rückgabewert Der Rückgabewert bezeichnet den Gerätekontext des vorgegebenen Fensters.

Hinweis Befindet sich das Caret im neu zu zeichnenden Bereich, wird es von *BeginPaint* verborgen, um ein Löschen zu vermeiden.

Querverweis *EndPaint, InvalidateRect, InvalidateRgn, WM_ERASEBKGND, WM_PAINT*

BitBlt

Syntax
```
BOOL BitBlt(HDC DestDC, int X, int Y, int Width,
int Height, HDC SrcDC, int XSrc, int YSrc, DWORD Rop);
```

Beschreibung Diese Funktion bewegt ein Bitmap von dem durch *SrcDC* bezeichneten Quellgerät zu dem durch *DestDC* vorgegebenen

Zielgerät. Die Parameter *XSrc* und *YSrc* bestimmen den Ursprung des Bitmap im Quellgerät. Die Parameter *X*, *Y*, *nWidth* und *Height* legen Ursprung, Breite und Höhe des durch das Bitmap auszufüllenden Rechtecks im Zielgerät fest. Der Parameter *Rop* definiert, wie die Bits von Quelle und Ziel kombiniert werden.

DestDC: Bezeichnet den Gerätekontext, der das Bitmap aufnehmen soll.

X: Bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Zielrechtecks.

Y: Bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Zielrechtecks.

Width: Bestimmt die Breite des Zielrechtecks und des Quell-Bitmaps in virtuellen Einheiten.

Height: Bestimmt die Höhe des Zielrechtecks und des Quell-Bitmaps in virtuellen Einheiten.

SrcDC: Bezeichnet den Gerätekontext, aus dem das Bitmap kopiert wird. Er muß den Wert 0 haben, wenn der Parameter *Rop* eine Rasteroperation festlegt, die keine Quelle einschließt.

XSrc: Bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Quell-Bitmap.

YSrc: Bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Quell-Bitmap.

Rop: Bestimmt die durchzuführende Rasteroperation.

Die Codes der Rasteroperationen legen fest, wie die grafische Geräteschnittstelle (GDI) Farben in Ausgaben kombiniert, in die der aktuelle Pinsel, ein mögliches Quell-Bitmap und ein Ziel-Bitmap einbezogen sind. Eine Liste der Codes der Rasteroperationen findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »Ternäre Rasteroperationen«.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Bitmap gezeichnet wurde, andernfalls ist er Null.

Hinweis Die GDI transformiert die Parameter *Width* und *Height*, indem sie einmal den Zielgerätekontext und einmal den Quellgerätekontext verwendet. Entsprechen die berechneten Größen einander nicht, verwendet die GDI die Funktion *StretchBlt*, um das Quell-Bitmap auf die entsprechenden Dimensionen zu bringen. Haben Ziel-, Quell- und Vorgabe-Bitmap unterschiedliche Farbformate, konvertiert die Funktion *BitBlt* Quell- und Vorgabe-Bitmap, um sie dem Ziel anzupassen. Für die Konvertierung werden die Vorder- und Hintergrundfarben des Ziels verwendet.

Konvertiert *BitBlt* monochrome in farbige Bitmaps, so werden weiße Bits (1) auf die Hintergrundfarbe und schwarze Bits (0) auf die Vordergrundfarbe gesetzt. Hierbei werden die Vorder- und Hintergrundfarben des Zielgerätekontextes verwendet. Bei der Konvertierung von farbigen in monochrome Bitmaps setzt *BitBlt* Pixel in der Hintergrundfarbe auf Weiß (1), alle anderen auf Schwarz (0). Hierbei werden die Vorder- und Hintergrundfarben des farbigen Quellgerätekontextes verwendet.

Vordergrundfarbe ist die aktuelle Textfarbe, Hintergrundfarbe die aktuelle Hintergrundfarbe des vorgegebenen Gerätekontextes.

Nicht alle Geräte unterstützen die Funktion *BitBlt*. Weitere Informationen finden sich in der Beschreibung der Funktion *GetDeviceCaps* und in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »Gerätefähigkeiten«.

BringWindowToTop

Syntax
```
void BringWindowToTop(HWND Wnd);
```

Beschreibung Diese Funktion bringt ein untergeordnetes Pop-up-Fenster in den Vordergrund eines Stapels überlappender Fenster. Zusätzlich werden Pop-up-Fenster und Hauptfenster aktiviert. *BringWindowToTop* kann zur Freilegung jedes Fensters verwendet werden, das ganz oder teilweise durch überlappende Fenster verdeckt wird.

Wnd: Bezeichnet das untergeordnete Pop-up-Fenster, das in den Vordergrund gebracht werden soll.

Rückgabewert Keiner.

BuildCommDCB

Syntax
```
int BuildCommDCB(LPSTR Def, DCB Far* DCBStruct);
```

Beschreibung Diese Funktion übersetzt den durch *Def* bezeichneten Definitions-String in entsprechende Gerätesteuerblockcodes und positioniert diese Codes in der durch *DCBStruct* bezeichneten Struktur.

Def: Zeigt auf einen nullterminierten String, der Steuerinformationen für ein Gerät enthält. Der String muß das Format der Parameter des DOS-Befehls MODE haben.

DCBStruct: Zeigt auf eine Datenstruktur *DCB*, die den übersetzten String aufnehmen soll. Die Struktur definiert die Steuerungsvoreinstellungen eines Geräts an der seriellen Schnittstelle.

Rückgabewert Der Rückgabewert ist 0, wenn der String übersetzt wurde und negativ, wenn ein Fehler auftritt.

Hinweis *BuildCommDCB* füllt nur den Puffer. Um diese Voreinstellungen an den Anschluß weiterzuleiten, sollten Anwendungen die Funktion *SetCommState* aufrufen. Als Vorgabe stellt *BuildCommDCB* Xon/Xoff ein und deaktiviert die Hardware-Steuerung. Außerdem sollten Anwendungen die entsprechenden Felder in der Datenstruktur *DCB* setzen, um die Übertragungssteuerung zu aktivieren.

Querverweis *SetCommState*

CallMsgFilter

Syntax
```
BOOL CallMsgFilter(LPMSG Msg, int Code);
```

Beschreibung Diese Funktion übergibt eine Botschaft und einen Code an die aktuelle Botschaftsfilterfunktion. Die Botschaftsfilterfunktion ist eine anwendungsspezifische Funktion, die alle Botschaften prüft und verändert. Sie wird durch Aufruf der Funktion *SetWindowsHook* festgelegt.

Msg: Zeigt auf eine Datenstruktur *MSG*, die die zu filternde Botschaft enthält.

Code: Bestimmt einen Code, der von der Filterfunktion verwendet wird, um zu entscheiden, wie sie die Botschaft bearbeiten soll.

Rückgabewert Der Rückgabewert ist Null, wenn die Botschaft bearbeitet und ungleich Null, wenn sie nicht mehr bearbeitet werden soll.

Hinweis *CallMsgFilter* wird normalerweise von Windows aufgerufen, um Anwendungen die Prüfung und Kontrolle des Botschaftsflusses zu ermöglichen, wenn ein Fenster verschoben oder in der Größe verändert wird bzw. während interner Abläufe in Menüs und Bildlaufleisten.

Werte, die im Parameter *Code* vorgegeben werden, dürfen sich nicht mit den von Windows an die Botschaftsfilterfunktion übergebenen *MSGF_*- und *HC_*-Werten überschneiden.

Querverweis *SetWindowsHook*

CallWindowProc

Syntax
```
LONG CallWindowProc(FARPROC PrevWndProc, HWND Wnd,
WORD Msg, WORD wParam, LONG lParam);
```

Beschreibung Diese Funktion übergibt Botschaftsinformationen an die Funktion, die durch den Parameter *PrevWndFunc* bezeichnet wird.

Sie wird für die Fensterunterklassifizierung benutzt. Norma-

lerweise nutzen alle Fenster mit der gleichen Klasse dieselbe Fensterfunktion gemeinsam. Eine Unterklasse ist ein Fenster oder eine Menge von Fenstern, die zur selben Fensterklasse gehören, deren Botschaften aber von einer oder mehreren anderen Funktion(en) abgefangen und bearbeitet werden, bevor sie an die Fensterfunktion dieser Klasse übergeben werden.

Die Funktion *SetWindowLong* erzeugt die Unterklasse, indem sie die einem bestimmten Fenster zugeordnete Fensterfunktion austauscht und Windows dazu veranlaßt, die neue Fensterfunktion anstelle der vorherigen aufzurufen. Jede Botschaft, die nicht von der neuen Fensterfunktion bearbeitet wird, muß an die alte Fensterfunktion durch den Aufruf von *CallWindowProc* übergeben werden. Dies erlaubt es, eine Kette von Fensterfunktionen zu bilden.

PrevWndProc: Ist die Adresse der Prozedurinstanz der vorherigen Fensterfunktion.

Wnd: Bezeichnet das Fenster, das die Botschaft empfängt.

Msg: Bestimmt die Botschaftsnummer.

wParam: Bestimmt zusätzliche botschaftsabhängige Informationen.

lParam: Bestimmt zusätzliche botschaftsabhängige Informationen.

Rückgabewert Es wird das Ergebnis der Bearbeitung der Botschaft zurückgeliefert. Die möglichen Rückgabewerte hängen von der gesendeten Botschaft ab.

Querverweis *SetWindowLong*

Catch

Syntax `int Catch(LPCATCHBUF CatchBuf);`

Beschreibung Diese Funktion erfaßt die momentane Ausführungsumgebung und kopiert die Daten in den Puffer, auf den der Parameter *CatchBuf* zeigt. Die Ausführungsumgebung ist der Zustand aller Systemregister und des Befehlszählers.

CatchBuf: Zeigt auf die *CATCHBUF*-Struktur, die die Ausführungsumgebung aufnimmt.

Rückgabewert Der Rückgabewert ist 0, wenn die Umgebung in den Puffer kopiert wurde.

Hinweis Die Funktion *Throw* verwendet den Puffer zur Wiederherstellung der Ausführungsumgebung mit den vorherigen Werten.

Die Funktion *Catch* gleicht der C-Laufzeitfunktion *setjmp*. *setjmp* wird von Windows nicht unterstützt.

Querverweis *Throw*

ChangeClipboardChain

Syntax
```
BOOL ChangeClipboardChain(HWND Wnd, HWND WndNext);
```

Beschreibung Diese Funktion entfernt das durch *Wnd* bezeichnete Fenster aus der Kette der Clipboard-Viewer und macht das durch *WndNext* bezeichnete Fenster zu dessen Nachfolger.

Wnd: Bezeichnet das Fenster, das aus der Kette entfernt werden soll. Das Handle muß vorher an die Funktion *SetClipboardViewer* übergeben worden sein.

WndNext: Bezeichnet das Fenster, das *Wnd* in der Kette der Clipboard-Viewer folgt. Das ist das Handle, das von der Funktion *SetClipboardViewer* zurückgegeben wird, sofern die Reihenfolge nicht infolge einer *WM_CHANGECBCHAIN*-Botschaft geändert wurde.

Rückgabewert Der Rückgabewert gibt den Status des durch *Wnd* bezeichneten Fensters an. Er ist ungleich Null, wenn das Fenster gefunden und entfernt wurde, andernfalls ist er Null.

Querverweis *SetClipboardViewer, WM_CHANGECBCHAIN*

CheckDlgButton

Syntax
```
void CheckDlgButton(HWND Dlg, int IDButton, WORD Check);
```

Beschreibung Diese Funktion markiert einen Schalter, entfernt eine Auswahlmarkierung oder ändert den Zustand eines dreistufigen Schalters. *CheckDlgButton* sendet eine *BM_SETCHECK*-Botschaft an den Schalter, der die angegebene ID im gegebenen Dialogfenster hat.

Dlg: Bezeichnet das Dialogfenster, das den Schalter enthält.

IDButton: Bestimmt den Schalter, der modifiziert werden soll.

Check: Gibt die vorzunehmende Handlung an. Ist der Parameter *Check* ungleich 0, markiert die Funktion *CheckDlgButton* den Schalter, ist er 0, wird die Auswahlmarkierung entfernt. Bei dreistufigen Schaltern wird der Schalter im nichtaktivierbaren Zustand dargestellt, wenn *Check* den Wert 2 hat, er wird markiert, wenn *Check* 1 ist und die Markierung wird entfernt, wenn *Check* 0 ist.

Rückgabewert Keiner.

CheckMenuItem

Syntax

```
BOOL CheckMenuItem(HMENU Menu, WORD IDCheckItem,
WORD Check);
```

Beschreibung Diese Funktion plaziert eine Auswahlmarkierung neben einer Menüoption oder entfernt sie von einer Menüoption in dem durch den Parameter *Menu* bezeichneten Pop-up-Menü. Der Parameter *IDCheckItem* bestimmt die Option, die verändert werden soll.

Menu: Bezeichnet das Menü.

IDCheckItem: Bestimmt die Menüoption, die bearbeitet werden soll.

Check: Bestimmt, wie die Menüoption markiert und die Position der Option im Menü festgelegt werden soll. Der Parameter *Check* kann eine Kombination der Flags *MF_CHECKED* oder *MF_UNCHECKED* mit *MF_BYPOSITION* oder *MF_BYCOMMAND* sein. Diese Flags können durch Verwendung des bitweisen ODER-Operators (|) kombiniert werden.

Wert	Bedeutung
MF_BYCOMMAND	Der Parameter *IDCheckItem* bestimmt die ID der Menüoption (Voreinstellung).
MF_BYPOSITION	Der Parameter *IDCheckItem* bestimmt die Position der Menüoption (die erste Menüoption liegt an der Position 0).
MF_CHECKED	Setzt die Auswahlmarkierung.
MF_UNCHECKED	Entfernt die Auswahlmarkierung.

Rückgabewert Der Rückgabewert gibt den momentanen Zustand der Option an, dieser ist entweder *MF_CHECKED* oder *MF_UNCHECKED*.

Existiert die Menüoption nicht, ist der Rückgabewert -1.

Hinweis Der Parameter *IDCheckItem* kann eine Pop-up-Menüoption oder eine Menüoption bezeichnen. Es sind keine speziellen Schritte nötig, um eine Pop-up-Menüoption zu markieren.

Hauptmenüoptionen können nicht markiert werden. Eine Pop-up-Menüoption muß durch Übergabe ihrer Position markiert werden, da sie keinen zugeordneten Bezeichner hat.

CheckRadioButton

Syntax

```
void CheckRadioButton(HWND Dlg, int IDFirstButton,
int IDLastButton, int IDCheckButton);
```

Beschreibung Diese Funktion markiert das durch den Parameter *IDCheckButton* gegebene Schaltfeld und entfernt die Auswahlmarkierung

von allen anderen Schaltfeldern in der durch die Parameter *IDFirstButton* und *IDLastButton* definierten Gruppe von Schaltern.

Die Funktion *CheckRadioButton* sendet eine *BM_SETCHECK*-Botschaft an das Schaltfeld, das die angegebene ID im gegebenen Dialogfenster hat.

Dlg: Bezeichnet das Dialogfenster.

IDFirstButton: Bestimmt den Integer-Bezeichner des ersten Schaltfelds in der Gruppe.

IDLastButton: Bestimmt den Integer-Bezeichner des letzten Schaltfelds in der Gruppe.

IDCheckButton: Bestimmt den Integer-Bezeichner des Schaltfelds, das markiert werden soll.

Rückgabewert Keiner.

ChildWindowFromPoint

Syntax
```
HWND ChildWindowFromPoint(HWND WndParent, POINT APoint);
```

Beschreibung Diese Funktion überprüft, welches der dem Fenster *Wnd* untergeordnete Fenster den angegebenen Punkt enthält.

WndParent: Bezeichnet das übergeordnete Fenster.

APoint: POINT-Struktur mit den Client-Koordinaten des Punktes, der getestet werden soll.

Rückgabewert Der Rückgabewert bezeichnet das untergeordnete Fenster, das den Punkt enthält. Er ist 0, wenn der übergebene Punkt außerhalb des übergeordneten Fensters liegt. Wenn der Punkt innerhalb des übergeordneten Fensters, aber nicht in einem der untergeordneten Fenster liegt, wird das Handle des übergeordneten Fensters zurückgegeben.

Chord

Syntax
```
BOOL Chord(HDC DC, int X1, int Y1, int X2, int Y2,
int X3, int Y3, int X4, int Y4);
```

Beschreibung Diese Funktion zeichnet einen gefüllten Ellipsenausschnitt (einen Bereich, der durch den Schnitt einer Ellipse und eines Liniensegments eingegrenzt wird). Die Parameter (*X1*, *Y1*) und (*X2*, *Y2*) bestimmen die obere linke bzw. untere rechte Ecke eines Rechtecks, das die vollständige Ellipse genau umgibt. Der Mittelpunkt des Rechtecks ist der Mittelpunkt der Ellipse. Die Parameter (*X3*, *Y3*) und (*X4*, *Y4*) geben die Endpunkte einer Linie an, die die Ellipse schneidet.

DC: Bezeichnet den Gerätekontext, in dem der elliptische Bogen dargestellt wird.

X1, Y1: Bestimmt die x- bzw. y-Koordinate der oberen linken Ecke des eingrenzenden Rechtecks.

X2, Y2: Bestimmt die x- bzw. y-Koordinate der unteren rechten Ecke des eingrenzenden Rechtecks.

X3, Y3: Bestimmt die x- bzw. y-Koordinate des ersten Endpunkts des Liniensegments.

X4, Y4: Bestimmt die x- bzw. y-Koordinate des zweiten Endpunkts des Liniensegments.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der Bogen gezeichnet wurde, andernfalls ist er Null.

Querverweis *Arc*

ClearCommBreak

Syntax
```
int ClearCommBreak(int Cid);
```

Beschreibung Diese Funktion stellt die zeichenweise Übertragung wieder her und versetzt die Übertragungsleitung in einen nicht zu unterbrechenden Zustand.

Cid: Bestimmt die Schnittstelle, die wiederhergestellt werden soll. Die Funktion *OpenComm* liefert diesen Wert zurück.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich ausgeführt wurde. Er ist negativ, wenn der Parameter *Cid* keine gültige Geräteeinheit bezeichnet.

Querverweis *OpenComm*

ClientToScreen

Syntax
```
void ClientToScreen(HWND Wnd, LPPOINT Point);
```

Beschreibung Diese Funktion wandelt die Client-Koordinaten eines gegebenen Punktes auf dem Bildschirm in Bildschirmkoordinaten um.

Sie verwendet die Client-Koordinaten der *POINT*-Datenstruktur, auf die der Parameter *Point* zeigt, um neue Bildschirmkoordinaten zu errechnen. Anschließend ersetzt sie die Koordinaten in der Struktur durch die neuen Koordinaten. Die neuen Bildschirmkoordinaten sind relativ zur oberen linken Ecke der Systemanzeige.

Wnd: Bezeichnet das Fenster, dessen Client-Bereich für die Umwandlung verwendet wird.

Point: Zeigt auf eine *POINT*-Datenstruktur, die die umzuwandelnden Client-Koordinaten enthält.

Rückgabewert Keiner.

Hinweis *ClientToScreen* setzt voraus, daß der übergebene Punkt in Client-Koordinaten und relativ zum vorgegebenen Fenster angegeben wird.

ClipCursor

Syntax
```
void ClipCursor(LPRECT Rect);
```

Beschreibung Diese Funktion schränkt den Mauszeiger auf das durch den Parameter *Rect* bezeichnete Rechteck auf dem Bildschirm ein. Liegt eine spätere Mauszeigerposition, die durch *SetCursorPos* oder die Maus selbst vorgegeben wird, außerhalb des Rechtecks, paßt Windows die Position automatisch an, um den Mauszeiger innerhalb des Rechtecks zu halten. Ist *Rect* NULL, kann sich der Mauszeiger auf dem ganzen Bildschirm bewegen.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die Bildschirmkoordinaten der oberen linken und der unteren rechten Ecke des eingrenzenden Rechtecks enthält.

Rückgabewert Keiner.

Hinweis Der Mauszeiger ist eine gemeinsam genutzte Ressource. Eine Anwendung, die den Mauszeiger auf ein gegebenes Rechteck beschränkt hat, muß ihn wieder freigeben, bevor sie die Steuerung einer anderen Anwendung überläßt.

Querverweis *SetCursorPos*

CloseClipboard

Syntax
```
BOOL CloseClipboard(void);
```

Beschreibung Diese Funktion schließt die Zwischenablage. Sie sollte aufgerufen werden, wenn ein Fenster das Prüfen oder Ändern der Zwischenablage beendet hat. Sie ermöglicht dadurch anderen Anwendungen den Zugriff auf die Zwischenablage.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Zwischenablage geschlossen wurde, andernfalls ist er Null.

CloseComm

Syntax
```
int CloseComm(int Cid);
```

Beschreibung Diese Funktion schließt die durch den Parameter *Cid* angegebene Schnittstelle und gibt den Speicher frei, der für die

Übertragungs- und Empfangs-Warteschlangen der Geräteeinheit belegt wurde. Alle Zeichen in der Ausgabe-Warteschlange werden gesendet, bevor die Schnittstelle geschlossen wird.

Cid: Bestimmt die Geräteeinheit, deren Schnittstelle geschlossen werden soll. Die Funktion *OpenComm* gibt diesen Wert zurück.

Rückgabewert Der Rückgabewert ist 0, wenn die Schnittstelle geschlossen wurde und negativ, wenn ein Fehler aufgetreten ist.

Querverweis *OpenComm*

CloseMetaFile

Syntax
```
HANDLE CloseMetaFile(HANDLE DC);
```

Beschreibung Diese Funktion schließt den Metafile-Gerätekontext und erzeugt ein Metafile-Handle, das verwendet werden kann, um das Metafile mit der Funktion *PlayMetaFile* abzuspielen.

DC: Bezeichnet den Metafile-Gerätekontext, der geschlossen werden soll.

Rückgabewert Der Rückgabewert bezeichnet das Metafile, wenn die Funktion erfolgreich war, andernfalls ist er 0.

Querverweis *PlayMetaFile*

CloseSound

Syntax
```
void CloseSound(void);
```

Beschreibung Diese Funktion schließt den Zugriff auf den Lautsprecher und gibt die Geräteeinheit für das Öffnen durch andere Anwendungen frei. Sie löscht alle Warteschlangen und gibt jeden hierfür reservierten Puffer frei. Die Funktion hat keine Parameter.

Rückgabewert Keiner.

CloseWindow

Syntax
```
void CloseWindow(HWND Wnd);
```

Beschreibung Diese Funktion verkleinert das angegebene Fenster. Wenn das Fenster andere überlappt, werden Client-Bereich und Überschrift des geöffneten Fensters vom Bildschirm entfernt und das Symbol in den Symbolbereich des Bildschirms verschoben.

Wnd: Bezeichnet das Fenster, das verkleinert werden soll.

Rückgabewert Keiner.

Hinweis Diese Funktion hat keine Wirkung, wenn der Parameter *Wnd* das Handle eines Pop-up- oder eines untergeordneten Fensters ist.

CombineRgn

Syntax
```
int CombineRgn(HRGN DestRgn, HRGN SrcRgn1, HRGN
SrcRgn2, int CombineMode);
```

Beschreibung Diese Funktion erzeugt eine neue Region, indem sie zwei existierende Regionen auf die durch den Parameter *CombineMode* angegebene Weise verbindet.

DestRgn: Bezeichnet eine Region, die durch die neue Region ersetzt wird.

SrcRgn1: Bezeichnet eine bestehende Region.

SrcRgn2: Bezeichnet eine bestehende Region.

CombineMode: Bestimmt die Operation, die auf die beiden bestehenden Regionen angewandt werden soll. Er kann jeden der folgenden Werte annehmen:

Wert	Bedeutung
RGN_AND	Verwendet nur die überlappenden Bereiche beider Regionen (Schnittmenge).
RGN_COPY	Erzeugt eine Kopie von Region 1 (*SrcRgn1*).
RGN_DIFF	Speichert die Bereiche von Region 1 (*SrcRgn1*), die nicht Teil von Region 2 (*SrcRgn2*) sind.
RGN_OR	Verbindet beide Regionen (Vereinigung).
RGN_XOR	Verbindet beide Regionen, entfernt aber überlappende Bereiche.

Rückgabewert Der Rückgabewert bestimmt den Typ der entstandenen Region. Folgende Werte sind möglich:

Wert	Bedeutung
COMPLEXREGION	Neue Region hat überlappende Ränder.
ERROR	Keine neue Region erzeugt.
NULLREGION	Neue Region ist leer.
SIMPLEREGION	Neue Region hat keine überlappenden Ränder.

Hinweis Falls der Parameter *DestRgn* keine bestehende Region bezeichnet, muß die Anwendung stattdessen einen **far**-Zeiger zu einer vorher erzeugten *HRGN* übergeben.

CopyMetaFile

Syntax
```
HANDLE CopyMetaFile(HANDLE SrcMetaFile, LPSTR FileName);
```

Beschreibung Diese Funktion kopiert das Quell-Metafile in die Datei, auf die der Parameter *FileName* zeigt und gibt ein Handle für das neue Metafile zurück. Wenn *FileName* NULL ist, wird die Quelldatei in ein Speicher-Metafile kopiert.

SrcMetaFile: Bezeichnet das Quell-Metafile.

FileName: Zeigt auf einen nullterminierten String, der die Datei angibt, die das Metafile aufnehmen soll.

Rückgabewert Der Rückgabewert bezeichnet das neue Metafile.

CopyRect

Syntax
```
int CopyRect(LPRECT DestRect, LPRECT SourceRect);
```

Beschreibung Diese Funktion kopiert das Rechteck, auf das der Parameter *SourceRect* zeigt, in die *RECT*-Datenstruktur, auf die der Parameter *DestRect* zeigt.

DestRect: Zeigt auf eine *RECT*-Datenstruktur.

SourceRect: Zeigt auf eine *RECT*-Datenstruktur.

Rückgabewert Der Rückgabewert dieser Funktion ist ohne Bedeutung.

CountClipboardFormats

Syntax
```
int CountClipboardFormats(void);
```

Beschreibung Diese Funktion liefert die Anzahl der Formate, die von der Zwischenablage verarbeitet werden können.
Die Funktion hat keine Parameter.

Rückgabewert Die Funktion liefert die Anzahl von Datenformaten in der Zwischenablage zurück.

CountVoiceNotes

Syntax
```
int CountVoiceNotes(int Voice);
```

Beschreibung Diese Funktion liefert die Anzahl der Noten in der angegebenen Klangwarteschlange. Es werden nur diejenigen Einträge in der Warteschlange gezählt, die aus einem Aufruf der Funktion *SetVoiceNote* hervorgegangen sind.

Voice: Bestimmt die Klangwarteschlange, deren Noten gezählt werden sollen. Die Nummer der ersten Warteschlange ist 1.

Rückgabewert Es wird die Anzahl der Noten in der vorgegebenen Klangwarteschlange zurückgegeben.

Querverweis *SetVoiceNote*

CreateBitmap

Syntax
```
HBITMAP CreateBitmap(int Width, int Height, BYTE
Planes, BYTE BitCount, LPSTR Bits);
```

Beschreibung Diese Funktion erzeugt ein geräteabhängiges Bitmap mit der angegebenen Breite und Höhe sowie dem vorgegebenen Bitmuster. Das Bitmap kann anschließend über die Funktion *SelectObject* als aktuelles Bitmap ausgewählt werden.

Obwohl ein Bitmap nicht direkt in ein Anzeigegerät kopiert werden kann, kann *BitBlt* es aus einem Speicheranzeigekontext (in dem es das aktuelle Bitmap ist) in jedes andere kompatible Gerät kopieren.

Width: Bestimmt die Breite des Bitmap in Pixel.

Height: Bestimmt die Höhe des Bitmap in Pixel.

Planes: Bestimmt die Anzahl der Farbebenen im Bitmap. Jede Ebene besitzt *Width* * *Height* * *BitCount* Bits.

BitCount: Bestimmt die Anzahl von Farbbits je Bildschirmpixel.

Bits: Zeigt auf ein Byte-Array, das die ursprünglichen Bitwerte des Bitmap enthält. Wenn Bits NULL ist, wird das neue Bitmap nicht initialisiert. Weitere Informationen finden sich in der Beschreibung des Feldes *bmBits* der *BITMAP*-Datenstruktur.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt wurde, bezeichnet der Rückgabewert ein Bitmap, andernfalls ist er NULL.

Querverweis *BitBlt, SelectObject*

CreateBitmapIndirect

Syntax
```
HBITMAP CreateBitmapIndirect(BITMAP FAR * Bitmap);
```

Beschreibung Diese Funktion erzeugt ein Bitmap, dessen Breite, Höhe und Bitmuster durch die Datenstruktur, auf die der Parameter *Bitmap* zeigt, vorgegeben werden.

Obwohl ein Bitmap nicht direkt für einen Bildschirm ausgewählt werden kann, kann es als aktuelles Bitmap gewählt und mit der Funktion *BitBlt* in jedes kompatible Anzeigegerät kopiert werden.

Bitmap: Zeigt auf eine *BITMAP*-Datenstruktur, die Informationen über das Bitmap enthält.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt wurde, bezeichnet der Rückgabewert ein Bitmap, andernfalls ist er NULL.

Querverweis *BitBlt*

CreateBrushIndirect

Syntax
```
HBRUSH CreateBrushIndirect(LOGBRUSH FAR * LogBrush);
```

Beschreibung Diese Funktion erzeugt einen virtuellen Pinsel, dessen Stil, Farbe und Muster durch die Datenstruktur vorgegeben werden, auf die der Parameter *LogBrush* zeigt. Der Pinsel kann anschließend für jede Geräteeinheit als aktueller Pinsel ausgewählt werden.

LogBrush: Zeigt auf eine *LOGBRUSH*-Datenstruktur, die Informationen über den Pinsel enthält.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt wurde, bezeichnet der Rückgabewert einen virtuellen Pinsel, andernfalls ist er NULL.

Hinweis Ein Pinsel, der unter Verwendung eines monochromen (eine Ebene, ein Bit pro Pixel) Bitmap erzeugt wurde, wird mit der aktuellen Text- und Hintergrundfarbe dargestellt. Pixel, die durch ein auf 0 gesetztes Bit repräsentiert werden, werden in der aktuellen Textfarbe, Pixel, bei denen das Bit auf 1 gesetzt ist, werden in der aktuellen Hintergrundfarbe dargestellt.

CreateCaret

Syntax
```
void CreateCaret(HWND Wnd, HBITMAP ABitmap, int Width,
int Height);
```

Beschreibung Diese Funktion erzeugt eine neue Form für das System-Caret und weist dem angegebenen Fenster den Besitz des Caret zu. Die Form des Caret kann eine Linie, ein Block oder ein Bitmap sein. Sie wird durch den Parameter *ABitmap* bestimmt.

Wenn *ABitmap* ein Bitmap-Handle ist, werden die Parameter *Width* und *Height* ignoriert, das Bitmap bestimmt seine Breite und Höhe selbst. Das Bitmap-Handle muß vorher durch die Verwendung einer der Funktionen *CreateBitmap*, *CreateDIBitmap* oder *LoadBitmap* erzeugt worden sein.

Wenn *ABitmap* 0 oder 1 ist, bestimmen *Width* und *Height* die Breite und Höhe des Caret in virtuellen Einheiten. Die genaue Breite und Höhe in Pixel ist vom Abbildungsmodus des Fensters abhängig.

Wenn *Width* oder *Height* 0 ist, richten sich Breite bzw. Höhe des Caret nach der Breite oder Höhe des Fensterrandes des Systems. Die Verwendung der Breite oder der Höhe des Fen-

sterrandes garantiert, daß das Caret auf einem hochauflösenden Bildschirm sichtbar ist.

CreateCaret löscht eine bereits zugewiesene Caret-Form automatisch, ohne Rücksicht darauf zu nehmen, welches Fenster das Caret besitzt. Wenn das Caret erzeugt ist, ist es zunächst unsichtbar. Um es sichtbar zu machen, muß die Funktion *ShowCaret* aufgerufen werden.

Wnd: Bezeichnet das Fenster, welches das neue Caret besitzt.

ABitmap: Bezeichnet das Bitmap, das die Form des Caret festlegt. Wenn *Bitmap* 0 ist, ist das Caret schwarz; ist *Bitmap* 1, ist das Caret grau.

Width: Bestimmt die Breite des Caret in virtuellen Einheiten.

Height: Bestimmt die Höhe des Caret in virtuellen Einheiten.

Rückgabewert Keiner.

Hinweis Das System-Caret ist eine gemeinsam genutzte Ressource. Ein Fenster sollte ein Caret nur dann erzeugen, wenn es den Fokus hat oder aktiv ist. Es sollte das Caret durch einen Aufruf von *DestroyCaret* wieder löschen, bevor es den Fokus verliert oder inaktiv wird.

Die Breite oder Höhe des Systemfensterrandes kann durch Verwendung der Funktion *GetSystemMetrics* mit den Codes *SM_CXBORDER* und *SM_CYBORDER* ermittelt werden (vergleiche Abschnitt »SM_XXXX« in Kapitel 4 »Windows-Stile und -Konstanten«,).

Querverweis *CreateBitmap, CreateDIBitmap, DestroyCaret, GetSystemMetrics, LoadBitmap*

CreateCompatibleBitmap

Syntax
```
HBITMAP CreateCompatibleBitmap(HDC DC, int Width,
int Height);
```

Beschreibung Diese Funktion erzeugt ein Bitmap, das mit dem durch den Parameter *DC* angegebenen Gerätekontext kompatibel ist. Das Bitmap hat die gleiche Anzahl an Farbebenen oder das gleiche Bits-pro-Pixel-Format wie der angegebene Gerätekontext. Es kann als aktuelles Bitmap für jedes Speichergerät, das mit dem von *DC* angegebenen kompatibel ist, gewählt werden.

Wenn *DC* ein Speichergerätekontext ist, hat das zurückgelieferte Bitmap das gleiche Format wie das in diesem Gerätekontext momentan gewählte Bitmap. Ein Speichergerätekontext ist ein Speicherblock, der eine Bildschirmoberfläche darstellt. Er kann dazu verwendet werden, Bilder im Speicher vorzubereiten, bevor sie auf die eigentliche Bildschirmoberfläche der kompatiblen Geräteeinheit kopiert werden.

Wenn ein Speichergerätekontext erzeugt wird, wählt die GDI automatisch ein vordefiniertes monochromes Bitmap.

Da für einen farbigen Speichergerätekontext sowohl farbige als auch monochrome Bitmaps ausgewählt werden können, ist das Format des Bitmap, das von *CreateCompatibleBitmap* zurückgeliefert wird, nicht immer das gleiche. Das Format eines kompatiblen Bitmap für einen Gerätekontext, der kein Speicher ist, entspricht immer dem Format der Geräteeinheit.

DC: Bezeichnet den Gerätekontext.

Width: Bestimmt die Breite des Bitmap in Bits.

Height: Bestimmt die Höhe des Bitmap in Bits.

Rückgabewert Der Rückgabewert bezeichnet ein Bitmap, wenn die Funktion erfolgreich war, andernfalls ist er NULL.

CreateCompatibleDC

Syntax
```
HDC CreateCompatibleDC(HDC DC);
```

Beschreibung Diese Funktion erzeugt einen Speichergerätekontext, der kompatibel mit der durch den Parameter *DC* bezeichneten Geräteeinheit ist. Ein Speichergerätekontext ist ein Speicherblock, der eine Bildschirmoberfläche darstellt. Er kann dazu verwendet werden, Bilder im Speicher vorzubereiten, bevor sie auf die eigentliche Bildschirmoberfläche der kompatiblen Geräteeinheit kopiert werden. Wenn ein Speichergerätekontext erzeugt wird, wählt die GDI für ihn automatisch ein vordefiniertes monochromes 1-zu-1-Bitmap.

DC: Bezeichnet den Gerätekontext. Wenn *DC* NULL ist, so erzeugt die Funktion einen Speichergerätekontext, der kompatibel zum Systembildschirm ist.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt wurde, bezeichnet der Rückgabewert den neuen Speichergerätekontext, andernfalls ist er NULL.

Hinweis Diese Funktion dient zur Erzeugung kompatibler Gerätekontexte für Geräteeinheiten, die Rasteroperationen unterstützen. Weitere Informationen finden sich in der Beschreibung der Funktion *GetDeviceCaps* und in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »DC_XXXX«.

GDI-Ausgabefunktionen können nur dann mit einem Speichergerätekontext verwendet werden, wenn ein Bitmap für diesen Kontext erzeugt und ausgewählt wurde.

Wenn eine Anwendung den Gerätekontext nicht länger benötigt, sollte sie ihn freigeben, indem sie die Funktion *DeleteDC* aufruft.

Querverweis *DeleteDC, GetDeviceCaps*

CreateCursor

Syntax
```
HCURSOR CreateCursor(HANDLE Instance, int XHotspot,
int YHotspot, int Width, int Height,LPSTR ANDBitPlane,
LPSTR XORBitPlane);
```

Beschreibung Diese Funktion erzeugt einen Mauszeiger mit der vorgegebenen Breite, Höhe und dem vorgegebenen Bitmuster.

Instance: Bezeichnet eine Instanz des Moduls, das den Mauszeiger erzeugt.

XHotspot: Bestimmt die horizontale Position des Aktionspunktes des Mauszeigers.

YHotspot: Bestimmt die vertikale Position des Aktionspunktes des Mauszeigers.

Width: Bestimmt die Breite des Mauszeigers in Pixeln.

Height: Bestimmt die Höhe des Mauszeigers in Pixeln.

ANDBitPlane: Zeigt auf ein Byte-Array, das die Bitwerte für die UND-Maske des Mauszeigers enthält. Dies können die Bits eines geräteabhängigen monochromen Bitmaps sein.

XORBitPlane: Zeigt auf ein Byte-Array, das die Bitwerte für die Exclusives-ODER-Maske des Mauszeigers enthält. Dies können die Bits eines geräteabhängigen monochromen Bitmaps sein.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt wurde, bezeichnet der Rückgabewert den Mauszeiger, andernfalls ist der Rückgabewert 0.

CreateDC

Syntax
```
HDC CreateDC(LPSTR DriverName, LPSTR DeviceName,
LPSTR Output, LPSTR InitData);
```

Beschreibung Diese Funktion erzeugt einen Gerätekontext für die angegebene Geräteeinheit. Die Parameter *DriverName*, *DeviceName* und *Output* geben den Gerätetreiber, den Gerätenamen und das physikalische Ausgabemedium an (Datei oder Schnittstelle).

DriverName: Zeigt auf einen nullterminierten String, der den DOS-Dateinamen (ohne Namenserweiterung) des Gerätetreibers enthält (zum Beispiel »Epson«).

DeviceName: Zeigt auf einen nullterminierten String, der den Namen der Geräteeinheit enthält, die unterstützt werden soll (zum Beispiel »Epson FX-80«). Der Parameter *DeviceName* wird verwendet, wenn das Modul mehr als eine Geräteeinheit unterstützt.

Output: Zeigt auf einen nullterminierten String, der den DOS-Dateinamen oder den Gerätenamen für das physikalische Ausgabemedium enthält (Datei oder Ausgabeanschluß).

InitData: Zeigt auf eine Windows-interne *DEVMODE*-Datenstruktur, die gerätespezifische Initialisierungsdaten für den Gerätetreiber enthält. Der Parameter *InitData* muß NULL sein, wenn der Gerätetreiber die eventuell vorhandene voreingestellte Initialisierung, die vom Anwender über die Systemsteuerung festgelegt wird, verwenden soll.

Rückgabewert Wenn die Funktion erfolgreich war, bezeichnet der Rückgabewert einen Gerätekontext für die angegebene Geräteeinheit, andernfalls ist er NULL.

Hinweis DOS-Gerätenamen folgen den DOS-Konventionen. Ein Doppelpunkt (:) am Ende ist empfehlenswert, jedoch optional. Windows entfernt den abschließenden Doppelpunkt, so daß ein Gerätename, der mit einem Doppelpunkt endet, dem gleichen Anschluß zugeordnet wird wie derselbe Name ohne einen Doppelpunkt. Die Treiber- und Anschlußnamen dürfen keine vorangestellten oder nachfolgenden Leerzeichen enthalten.

CreateDialog

Syntax
```
HWND CreateDialog(HANDLE Instance, LPSTR TemplateName,
HWND WndParent, FARPROC DialogFunc);
```

Beschreibung Diese Funktion erzeugt ein nicht-modales Dialogfenster, das die Größe, den Stil und die Steuerelemente besitzt, die durch die vom Parameter *TemplateName* bezeichnete Dialogfenstervorlage definiert werden. Der Parameter *WndParent* bezeichnet ein das Dialogfenster besitzendes Anwendungsfenster. Die Dialogfunktion, auf die der Parameter *DialogFunc* zeigt, bearbeitet jede Botschaft, die vom Dialogfenster empfangen wird.

CreateDialog sendet eine *WM_INITDIALOG*-Botschaft an die Dialogfunktion, bevor sie das Dialogfenster anzeigt. Diese Botschaft erlaubt es der Dialogfunktion, die Steuerelemente des Dialogfensters zu initialisieren.

CreateDialog kehrt sofort zurück, wenn sie das Dialogfenster erzeugt hat, sie wartet nicht, bis das Dialogfenster die Eingabebearbeitung beginnt.

Instance: Bezeichnet eine Instanz des Moduls, dessen ausführbare Datei die Vorlage des Dialogfensters enthält.

TemplateName: Zeigt auf einen nullterminierten String, der die Vorlage des Dialogfensters benennt.

WndParent: Bezeichnet das Fenster, welches das Dialogfenster besitzt.

DialogFunc: Ist die Adresse der Prozedurinstanz der Dialogfunktion. Details finden sich im folgenden »Hinweis«.

Rückgabewert Bei erfolgreicher Ausführung der Funktion ist der Rückgabewert das Fenster-Handle des Dialogfensters. Er ist NULL, wenn die Funktion das Dialogfenster nicht erzeugen kann.

Hinweis Verwenden Sie den Stil *WS_VISIBLE* in der Vorlage des Dialogfensters, wenn das Dialogfenster nach der Erzeugung im übergeordneten Fenster erscheinen soll.

Verwenden Sie die Funktion *DestroyWindow*, um ein Dialogfenster zu entfernen, das durch *CreateDialog* erzeugt wurde.

Ein Dialogfenster kann bis zu 255 Steuerelemente enthalten.

Callback-Funktion

```
BOOL FAR PASCAL DialogFunc(HWND Dlg, WORD Msg,
WORD wParam, DWORD lParam);
```

DialogFunc: Platzhalter für den Namen der von der Anwendung definierten Funktion. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Dlg: Bezeichnet das Dialogfenster, das die Botschaft empfängt.

Msg: Bestimmt die Nummer der Botschaft.

wParam: Bestimmt 16 Bits zusätzliche, botschaftsabhängige Informationen.

lParam: Bestimmt 32 Bits zusätzliche, botschaftsabhängige Informationen.

Rückgabewert Mit Ausnahme der Reaktion auf die *WM_INITDIALOG*-Botschaft sollte die Dialogfunktion einen Wert ungleich 0 zurückliefern, wenn die Funktion die Botschaft bearbeitet und 0, falls sie es nicht tut.
Als Antwort auf eine *WM_INITDIALOG*-Botschaft sollte die Dialogfunktion 0 zurückliefern, wenn sie die Funktion *SetFocus* aufruft, um den Fokus auf eines der Steuerelemente im Dialogfenster zu setzen. Andernfalls sollte sie einen Wert ungleich 0 zurückliefern. In diesem Fall wird Windows den Fokus auf das erste Steuerelement im Dialogfenster setzen, dem der Fokus gegeben werden kann.

Hinweis Die Dialogfunktion wird nur genutzt, wenn die Dialogklasse für das Dialogfenster verwendet wird. Dies ist die voreingestellte Klasse. Sie wird verwendet, wenn nicht ausdrücklich eine andere Klasse in der Dialogfenstervorlage vorgegeben wird.
Obwohl die Dialogfunktion einer Fensterfunktion sehr ähnlich ist, darf sie nicht die Funktion *DefWindowProc* aufrufen, um ungewollte Botschaften zu bearbeiten. Ungewollte Botschaften werden intern von der Fensterfunktion der Dialogklasse bearbeitet.

Die Adresse der Dialogfunktion, die als Parameter *DialogFunc* übergeben wird, muß durch Verwendung der Funktion *MakeProcInstance* erzeugt werden.

Querverweis *DefDlgProc, DestroyWindow, MakeProcInstance, WM_INITDIALOG*

CreateDialogIndirect

Syntax
```
HWND CreateDialogIndirect(HANDLE Instance,
LPSTR Template, HWND Parent, FARPROC DialogFunc);
```

Beschreibung Diese Funktion erzeugt ein nicht-modales Dialogfenster, das die Größe, den Stil und die Steuerelemente besitzt, die durch die vom Parameter *Template* bezeichnete Dialogfenstervorlage definiert werden.

Der Parameter *Parent* bezeichnet ein das Dialogfenster besitzendes Anwendungsfenster. Die Dialogfunktion, auf die der Parameter *DialogFunc* zeigt, bearbeitet jede Botschaft, die vom Dialogfenster empfangen wird.

CreateDialogIndirect sendet eine *WM_INITDIALOG*-Botschaft an die Dialogfunktion, bevor sie das Dialogfenster anzeigt. Diese Botschaft erlaubt es der Dialogfunktion, die Steuerelemente des Dialogfensters zu initialisieren.

CreateDialogIndirect kehrt sofort zurück, nachdem sie das Dialogfenster erzeugt hat. Sie wartet nicht, bis das Dialogfenster die Eingabebearbeitung beginnt.

Instance: Bezeichnet eine Instanz des Moduls, dessen ausführbare Datei die Vorlage des Dialogfenster enthält.

Template: Zeigt auf einen Speicherblock, der eine *DLGTEMPLATE*-Datenstruktur enthält.

Parent: Bezeichnet das Fenster, welches das Dialogfenster besitzt.

DialogFunc: Ist die Adresse der Prozedurinstanz der Dialogfunktion. Details finden sich im folgenden »Hinweis«.

Rückgabewert Der Rückgabewert ist bei erfolgreicher Ausführung der Funktion das Fenster-Handle des Dialogfensters. Er ist NULL, wenn die Funktion das Dialogfenster oder eines der Steuerelemente im Dialogfenster nicht erzeugen kann.

Hinweis Wird der Stil *WS_VISIBLE* in der Vorlage des Dialogfensters verwendet, so erscheint das Dialogfenster nach der Erzeugung im übergeordneten Fenster.

Ein Dialogfenster kann bis zu 255 Steuerelemente enthalten.

Callback-Funktion
```
BOOL FAR PASCAL DialogFunc(HWND Dlg, WORD Msg,
WORD wParam, DWORD lParam);
```

DialogFunc: Platzhalter für den Namen der anwendungsunterstützten Funktion. Der tatsächliche Name muß in der Moduldefinitionsdatei der Anwendung in einer EXPORTS-Anweisung exportiert werden.

Dlg: Bezeichnet das Dialogfenster, das die Botschaft empfängt.

Msg: Bezeichnet die Nummer der Botschaft.

wParam: Bezeichnet 16 Bits zusätzliche, botschaftsabhängige Informationen.

lParam: Bezeichnet 32 Bits zusätzliche, botschaftsabhängige Informationen.

Rückgabewert Mit Ausnahme der Reaktion auf die *WM_INITDIALOG*-Botschaft sollte die Dialogfunktion einen Wert ungleich 0 zurückliefern, wenn die Funktion die Botschaft bearbeitet und 0, falls sie es nicht tut.

Als Antwort auf eine *WM_INITDIALOG*-Botschaft sollte die Dialogfunktion 0 zurückliefern, wenn sie die Funktion *SetFocus* aufruft, um den Fokus auf eines der Steuerelemente im Dialogfenster zu setzen. Andernfalls sollte sie einen Wert ungleich 0 zurückliefern. In diesem Fall wird Windows den Fokus auf das erste Steuerelement im Dialogfenster setzen, dem der Fokus gegeben werden kann.

Hinweis Die Dialogfunktion wird nur genutzt, wenn die Dialogklasse für das Dialogfenster verwendet wird. Diese voreingestellte Klasse wird verwendet, wenn ausdrücklich keine andere Klasse in der Dialogfenstervorlage vorgegeben wird.

Obwohl die Dialogfunktion einer Fensterfunktion sehr ähnlich ist, darf sie nicht die Funktion *DefWindowProc* aufrufen, um ungewollte Botschaften zu bearbeiten. Ungewollte Botschaften werden intern von der Fensterfunktion der Dialogklasse bearbeitet.

Die Adresse der Dialogfunktion, die als Parameter *DialogFunc* übergeben wird, muß durch Verwendung der Funktion *MakeProcInstance* erzeugt werden.

Querverweis *DefDlgProc, MakeProcInstance, WM_INITDIALOG*

CreateDialogIndirectParam

Syntax
```
HWND CreateDialogIndirectParam(HANDLE Instance,
LPSTR Template, HWND WndParent, FARPROC DialogFunc,
LONG InitParam);
```

Beschreibung Diese Funktion erzeugt ein nicht-modales Dialogfenster, sendet eine *WM_INITDIALOG*-Botschaft an die Dialogfunktion, bevor sie das Dialogfenster anzeigt und übergibt *InitParam* als Parameter *lParam* der Botschaft. Diese Botschaft erlaubt es der Dialogfunktion, die Steuerelemente des Dialogfensters zu in-

itialisieren. Im übrigen ist diese Funktion identisch mit der Funktion *CreateDialogIndirect*.

Weitere Hinweise zum Erzeugen von nicht-modalen Dialogfenstern finden sich in der Beschreibung der Funktion *CreateDialogIndirect*.

Instance: Bezeichnet eine Instanz des Moduls, dessen ausführbare Datei die Vorlage des Dialogfensters enthält.

Template: Zeigt auf einen Speicherblock, der eine Datenstruktur vom Typ *DLGTEMPLATE* enthält.

WndParent: Bezeichnet das Fenster, welches das Dialogfenster besitzt.

DialogFunc: Ist die Adresse der Prozedurinstanz der Dialogfunktion. Einzelheiten sind in der Beschreibung der Funktion *CreateDialogIndirect* zu finden.

InitParam: Ein 32-Bit Wert, den *CreateDialogIndirectParam* der Dialogfunktion übergibt, wenn sie das Dialogfenster erzeugt.

Rückgabewert Der Rückgabewert ist bei erfolgreicher Ausführung der Funktion das Fenster-Handle des Dialogfensters. Er ist NULL, wenn die Funktion das Dialogfenster oder eines der Steuerelemente im Dialogfenster nicht erzeugen kann.

Querverweis *DefDlgProc, MakeProcInstance, WM_INITDIALOG*

CreateDialogParam

Syntax
```
HWND CreateDialogParam(HANDLE Instance,
LPSTR TemplateName, HWND WndParent, FARPROC DialogFunc,
LONG InitParam);
```

Beschreibung Diese Funktion erzeugt ein nicht-modales Dialogfenster, das die Größe, den Stil und die Steuerelemente besitzt, die durch die vom Parameter *TemplateName* bezeichnete Dialogfenstervorlage definiert werden. Der Parameter *WndParent* bezeichnet das Anwendungsfenster, welches das Dialogfenster besitzt. Die Dialogfunktion, auf die der Parameter *DialogFunc* zeigt, bearbeitet jede Botschaft, die vom Dialogfenster empfangen wird.

CreateDialogParam sendet die Botschaft *WM_INITDIALOG* an die Dialogfunktion, bevor das Dialogfenster angezeigt wird. Diese Botschaft ermöglicht die Initialisierung der Steuerelemente des Dialogfensters durch die Dialogfunktion. Als Parameter *lParam* der Botschaft wird *InitParam* verwendet. *CreateDialogParam* kehrt unmittelbar nach Erzeugung des Dialogfensters zurück. Der Beginn der Eingabebearbeitung wird nicht abgewartet.

Instance: Bezeichnet eine Instanz des Moduls, dessen ausführbare Datei die Vorlage des Dialogfensters enthält.

TemplateName: Zeigt auf einen nullterminierten String, der die Vorlage des Dialogfensters benennt.

Parent: Bezeichnet das Fenster, welches das Dialogfenster besitzt.

DialogFunc: Ist die Adresse der Prozedurinstanz der Dialogfunktion oder NULL, wenn diese durch die Klasse definiert ist.

InitParam: Wird als Parameter *lParam* der Botschaft *WM_INITDIALOG* übergeben, wenn das Dialogfenster erzeugt wird.

Rückgabewert Der Rückgabewert ist bei erfolgreicher Ausführung der Funktion das Fenster-Handle des Dialogfensters. Er ist NULL, wenn die Funktion das Dialogfenster nicht erzeugen kann.

CreateDIBitmap

Syntax
```
HBITMAP CreateDIBitmap(HDC DC, LPBITMAPINFOHEADER
InfoHeader, DWORD InitParam, LPSTR InitBits,
LPBITMAPINFO InitInfo, WORD Usage);
```

Beschreibung Diese Funktion erzeugt ein gerätespezifisches Speicher-Bitmap aus einer geräteunabhängigen Bitmap-Beschreibung (DIB) und setzt optional Bits im Bitmap.

DC: Bezeichnet den Gerätekontext.

InfoHeader: Zeigt auf eine *BITMAPINFOHEADER*-Struktur, die die Größe und das Format des geräteunabhängigen Bitmap beschreibt.

InitParam: Bestimmt, ob das Speicher-Bitmap initialisiert werden soll. Wenn *InitParam* auf *CBM_INIT* gesetzt ist, initialisiert *CreateDIBitmap* das Bitmap mit den durch *InitBits* und *InitInfo* bezeichneten Bits.

InitBits: Zeigt auf ein Byte-Array, das die ursprünglichen Bitmap-Werte enthält. Das Format der Bitmap-Werte hängt vom Feld *biBitCount* der *BITMAPINFOHEADER*-Struktur ab. Diese ist mit der durch *InitInfo* bezeichneten *BITMAPINFO*-Datenstruktur verknüpft.

InitInfo: Zeigt auf eine *BITMAPINFO*-Datenstruktur, die die Abmessungen und das Farbformat von *InitBits* beschreibt.

Usage: Bestimmt, ob die *bmiColors*-Elemente der *InitInfo*-Datenstruktur explizite RGB-Werte oder Indizes in die aktuell realisierte virtuelle Palette aufnehmen. Der Parameter *Usage* muß einen der folgenden Werte annehmen:

Wert	Bedeutung
DIB_PAL_COLORS	Die Farbtabelle besteht aus einem Array von

	16-Bit-Indizes in die aktuell realisierte virtuelle Palette.
DIB_RGB_COLORS	Die Farbtabelle enthält explizite RGB-Werte.

Rückgabewert Der Rückgabewert bezeichnet ein Bitmap, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls ist er NULL.

CreateDIBPatternBrush

Syntax
```
HBRUSH CreateDIBPatternBrush(HANDLE PackedDIB,
WORD Usage);
```

Beschreibung Diese Funktion erzeugt einen virtuellen Pinsel mit dem Muster, das von dem durch den Parameter *PackedDIB* bezeichneten geräteunabhängigen Bitmap (DIB) angegegeben wird. Der Pinsel kann später für jede Geräteeinheit ausgewählt werden, die Rasteroperationen unterstützt. Weitere Informationen finden sich in der Beschreibung der Funktion *GetDeviceCaps* und in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »Gerätefähigkeiten«.

PackedDIB: Bezeichnet ein globales Speicherobjekt, das ein gepacktes geräteunabhängiges Bitmap enthält. Um dieses Handle zu erhalten, ruft eine Anwendung die Funktion *GlobalAlloc* zur Reservierung eines Blocks im globalen Speichers auf und füllt dann den Speicher mit dem gepackten DIB. Ein gepacktes DIB besteht aus einer *BITMAPINFO*-Datenstruktur, unmittelbar gefolgt von einem Byte-Array, das die Pixel des Bitmap definiert.

Usage: Bestimmt, ob die *bmiColors*-Elemente der *BITMAPINFO*-Datenstruktur explizite RGB-Werte oder Indizes in die gegenwärtig realisierte virtuelle Palette enthalten.

Der Parameter *Usage* muß einen der folgenden Werte annehmen:

Wert	Bedeutung
DIB_PAL_COLORS	Die Farbtabelle besteht aus einem Array von 16-Bit-Indizes in die aktuell realisierte virtuelle Palette.
DIB_RGB_COLORS	Die Farbtabelle enthält explizite RGB-Werte.

Rückgabewert Der Rückgabewert bezeichnet einen virtuellen Pinsel, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Querverweis *GetDeviceCaps*

CreateDiscardableBitmap

Syntax
```
HBITMAP CreateDiscardableBitmap(HDC DC, int Width,
int Height);
```

Beschreibung Diese Funktion erzeugt ein Bitmap, dem das Attribut »discardable« zugeordnet ist und das mit dem durch den Parameter *DC* bezeichneten Gerätekontext kompatibel ist. Das Bitmap hat die gleiche Anzahl an Farbebenen bzw. das gleiche Bits-proPixel-Format wie die angegebene Geräteeinheit. Eine Anwendung kann dieses Bitmap als aktuelles Bitmap für ein Speichergerät auswählen, wenn es mit dem vom Parameter *DC* vorgegebenen kompatibel ist.

DC: Bezeichnet einen Gerätekontext.

Width: Bestimmt die Breite des Bitmap in Bits.

Height: Bestimmt die Höhe des Bitmap in Bits.

Rückgabewert Der Rückgabewert bezeichnet ein Bitmap, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls ist er NULL.

Hinweis Windows kann einem Bitmap, das von dieser Funktion erzeugt wurde, nur verwerfen, wenn es nicht durch eine Anwendung für einen Bildschirmkontext ausgewählt ist. Verwirft Windows das Bitmap, solange es nicht ausgewählt ist, und versucht die Anwendung später, es auszuwählen, wird die Funktion *SelectObject* eine 0 zurückliefern. In diesem Fall sollte die Anwendung das Handle des Bitmap durch Verwendung von *DeleteObject* entfernen.

CreateEllipticRgn

Syntax
```
HRGN CreateEllipticRgn(int X1, int Y1, int X2, int Y2);
```

Beschreibung Diese Funktion erzeugt eine elliptische Region.

X1, Y1: Bestimmt die x- bzw. y-Koordinate der oberen linken Ecke des umgebenden Rechtecks der Ellipse.

X2, Y2: Bestimmt die x- bzw. y-Koordinate der unteren rechten Ecke des umgebenden Rechtecks der Ellipse.

Rückgabewert Der Rückgabewert bezeichnet eine neue Region, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls ist er NULL.

Hinweis Die Breite des Rechtecks, die durch den absoluten Wert der Differenz *X2 - X1* gegeben ist, darf 32.767 Einheiten nicht überschreiten. Diese Beschränkung gilt auch für die Höhe des Rechtecks (*Y2 - Y1*).

CreateEllipticRgnIndirect

Syntax
```
HRGN CreateEllipticRgnIndirect(LPRECT Rect);
```

Beschreibung Diese Funktion erzeugt eine elliptische Region.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die Koordinaten der oberen linken und der unteren rechten Ecke des Begrenzungsrechtecks der Ellipse enthält.

Rückgabewert Der Rückgabewert bezeichnet eine neue Region, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Hinweis Die Breite des Rechtecks darf 32.767 Einheiten nicht überschreiten. Diese Beschränkung gilt auch für die Höhe des Rechtecks.

CreateFont

Syntax
```
HFONT CreateFont(int Height, int Width, int Escapement,
int Orientation, int Weight, BYTE Italic,
BYTE Underline, BYTE StrikeOut, BYTE CharSet,
BYTE OutputPrecision, BYTE ClipPrecision,
BYTE Quality, BYTE PitchAndFamily, LPSTR FaceName);
```

Beschreibung Diese Funktion erzeugt eine virtuelle Schrift, welche die angegebenen Merkmale hat. Die virtuelle Schrift kann später für jede Geräteeinheit als aktuelle Schrift ausgewählt werden.

Height: Gibt die gewünschte Höhe der Schrift in virtuellen Einheiten an. Die Schrifthöhe kann auf drei Arten angegeben werden:

- Wenn *Height* größer als 0 ist, wird der Wert in Geräteeinheiten umgewandelt und an die Zellenhöhe der verfügbaren Schriften angepaßt.
- Wenn *Height* 0 ist, wird eine angemessene voreingestellte Größe verwendet.
- Wenn *Height* kleiner als 0 ist, wird er in Geräteeinheiten umgewandelt und der absolute Wert an die Zellenhöhe der verfügbaren Schriften angepaßt.

Für alle Höhenvergleiche sucht der Schriftabbilder nach der höchsten Schrift, die die gewünschte Größe nicht überschreitet und – falls es eine solche Schrift nicht gibt – nach der kleinsten verfügbaren Schrift.

Width: Bestimmt die durchschnittliche Breite der Schrift in virtuellen Einheiten. Wenn *Width* 0 ist, wird das Höhen-/Seitenverhältnis der Geräteeinheit mit dem Höhen-/Seitenverhältnis der verfügbaren Schriften verglichen, um das passend-

ste Gegenstück, das durch den absoluten Wert der Differenz ermittelt wird, festzustellen.

Escapement: Bestimmt den Winkel jeder Textzeile, die in der Schrift geschrieben wird, in Zehntelgraden relativ zum unteren Seitenrand.

Orientation: Bestimmt den Winkel der unteren Grenze jedes Zeichens, in zehntel Grad relativ zum unteren Seitenrand.

Weight: Bestimmt die gewünschte Schriftstärke im Bereich von 0 bis 1000 (400 ist normal, 700 ist fett). Wenn *Weight* 0 ist, wird ein voreingestelltes Maß verwendet. Außerdem kann auf die in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »FW_XXXX« beschriebenen Werte zurückgegriffen werden.

Italic: Bestimmt, ob die Schrift kursiv ist.

Underline: Bestimmt, ob die Schrift unterstrichen ist.

StrikeOut: Bestimmt, ob die Zeichen der Schrift durchgestrichen dargestellt werden.

CharSet: Gibt den gewünschten Zeichensatz an. Die folgenden Werte sind vordefiniert: *ANSI_CHARSET*, *OEM_CHARSET* und *SYMBOL_CHARSET*.

Der OEM-Zeichensatz ist systemabhängig, Schriften mit anderen Zeichensätzen können im System existieren. Wenn eine Anwendung eine Schrift mit einem unbekannten Zeichensatz verwendet, sollte sie nicht versuchen, Strings zu übersetzen oder auszuwerten, die in dieser Schrift dargestellt werden sollen. Statt dessen sollten die Strings direkt an den Ausgabegerätetreiber übergeben werden.

OutputPrecision: Gibt die gewünschte Ausgabepräzision an. Diese bestimmt, wie exakt die gewünschte Höhe, Breite, Zeichenausrichtung sowie Flucht und Neigung der Schrift bei der Ausgabe dargestellt werden. Er kann jeden der folgenden Werte annehmen: *OUT_CHARACTER_PRECIS*, *OUT_DEFAULT_PRECIS*, *OUT_STRING_PRECIS* oder *OUT_STROKE_PRECIS*.

ClipPrecision: Gibt die gewünschte Clipping-Präzision an, die bestimmt, wie Zeichen abgeschnitten werden sollen, die teilweise außerhalb des Clipping-Bereiches liegen. Er kann jeden der folgenden Werte annehmen: *CLIP_CHARACTER_PRECIS*, *CLIP_DEFAULT_PRECIS* oder *CLIP_STROKE_PRECIS*.

Quality: Gib die gewünschte Ausgabequalität an, die bestimmt, wie sorgfältig die GDI versucht, die virtuellen Schriftattribute denen einer tatsächlichen physikalischen Schrift anzugleichen. Er kann jeden der folgenden Werte annehmen: *DEFAULT_QUALITY*, *DRAFT_QUALITY* oder *PROOF_QUALITY* (siehe Kapitel 4 »Windows-Stile und -Konstanten«).

PitchAndFamily: Gibt Zeichenabstand und Herkunft der Schrift an. Die zwei niederwertigen Bits bestimmen den Zeichenabstand der Schrift und können jeden der folgenden Werte annehmen: *DEFAULT_PITCH*, *FIXED_PITCH* bzw. *VARIABLE_PITCH*. Die vier höherwertigen Bits des Parameters bestimmen die Schriftenfamilie und können jeden der folgenden Werte annehmen: *FF_DECORATIVE*, *FF_DONTCARE*, *FF_MODERN*, *FF_ROMAN*, *FF_SCRIPT* oder *FF_SWISS*. Diese Konstanten sind in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »FF_XXXX« beschrieben.

Facename: Zeigt auf einen nullterminierten String, der den Typennamen der Schrift enthält. Die Länge dieses Strings darf 30 Zeichen nicht überschreiten. Die Funktion *EnumFonts* kann verwendet werden, um die Typennamen aller aktuell verfügbaren Schriften aufzulisten.

Rückgabewert Der Rückgabewert bezeichnet eine virtuelle Schrift, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Hinweis *CreateFont* erzeugt keine neue Schrift. Die Funktion wählt nur das passendste Gegenstück aus den Schriften aus, die in der physikalischen Schriftensammlung der GDI verfügbar sind.

Querverweis *EnumFonts*

CreateFontIndirect

Syntax
```
HFONT CreateFontIndirect(LOGFONT FAR * LogFont);
```

Beschreibung Diese Funktion erzeugt eine virtuelle Schrift mit den Merkmalen, die von der Datenstruktur, auf die der Parameter *LogFont* zeigt, vorgegeben werden. Die Schrift kann später von jeder Geräteeinheit als aktuelle Schrift ausgewählt werden.

LogFont: Zeigt auf eine *LOGFONT*-Datenstruktur, die die Merkmale der virtuellen Schrift bestimmt.

Rückgabewert Der Rückgabewert bezeichnet eine virtuelle Schrift, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Hinweis *CreateFontIndirect* erzeugt eine virtuelle Schrift, die alle angegebenen Merkmale besitzt. Wenn die Schrift mit der Funktion *SelectObject* ausgewählt wird, versucht der Schriftabbilder der GDI, eine zur virtuellen Schrift passende physikalische Schrift zu finden. Findet er keine exakt passende Schrift, stellt er eine

Alternative zur Verfügung, die so viele der gewünschten Merkmale wie möglich aufweist.

CreateHatchBrush

Syntax

```
HBRUSH CreateHatchBrush(int Index, COLORREF Color);
```

Beschreibung

Diese Funktion erzeugt einen virtuellen Pinsel mit der angegebenen Schraffur und Farbe. Der Pinsel kann anschließend von jeder Geräteeinheit als aktueller Pinsel ausgewählt werden.

Index: Bestimmt den Schraffurstil des Pinsels. Er kann jeden der folgenden Werte annehmen:

Wert	Bedeutung
HS_BDIAGONAL	45-Grad abwärts schraffiert (von links oben nach rechts unten) (\\\\\).
HS_CROSS	Horizontal und vertikal überkreuz schraffiert (+++++).
HS_*DIAGCROSS*	45-Grad überkreuz schraffiert (xxxxx).
HS_FDIAGONAL	45-Grad aufwärts schraffiert (von links unten nach rechts oben) (/////).
HS_HORIZONTAL	Horizontal schraffiert (-----).
HS_VERTICAL	Vertikal schraffiert (||||).

Color: Bestimmt die Vordergrundfarbe des Pinsels bzw. die Farbe der Schraffur.

Rückgabewert

Der Rückgabewert bezeichnet einen virtuellen Pinsel, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

CreateIC

Syntax

```
HDC CreateIC(LPSTR DriverName, LPSTR DeviceName,
LPSTR Output, LPSTR InitData);
```

Beschreibung

Diese Funktion erzeugt einen Informationskontext für die angegebene Geräteeinheit, der es ermöglicht, schnell Informationen über die Geräteeinheit zu erlangen, ohne einen Gerätekontext zu erzeugen.

DriverName: Zeigt auf einen nullterminierten String, der den OS-Dateinamen des Gerätetreibers ohne Namenserweiterung enthält (zum Beispiel »Epson«).

DeviceName: Zeigt auf einen nullterminierten String, der den Namen der Geräteeinheit enthält (z.B. »Epson FX-80«). Der Parameter *DeviceName* wird verwendet, wenn das Modul mehr als eine Geräteeinheit unterstützt.

Output: Zeigt auf einen nullterminierten String, der den DOS-

Dateinamen oder den Gerätenamen für das physikalische Ausgabemedium enthält (Datei oder Ausgabeanschluß).

InitData: Zeigt auf gerätespezifische Initialisierungsdaten für den Gerätetreiber. Der Parameter *InitData* muß NULL sein, wenn der Gerätetreiber die vom Anwender durch die Systemsteuerung voreingestellte Initialisierung (falls vorhanden) verwenden soll.

Rückgabewert Der Rückgabewert bezeichnet einen Informationskontext für die angegebene Geräteeinheit, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls ist er NULL.

Hinweis DOS-Gerätenamen folgen den DOS-Konventionen. Ein Doppelpunkt (:) am Ende ist empfehlenswert, jedoch optional. Windows entfernt den abschließenden Doppelpunkt, so daß ein Gerätename, der mit einem Doppelpunkt endet, dem gleichen Anschluß zugeordnet wird, wie der gleiche Name ohne einen Doppelpunkt. Die Treiber- und Anschlußnamen dürfen keine führenden oder abschließenden Leerzeichen enthalten. GDI-Ausgabefunktionen können nicht mit Informationskontexten benutzt werden.

CreateIcon

Syntax
```
HICON CreateIcon(HANDLE Instance, int Width,
int Height, BYTE Planes, BYTE BitsPixel, LPSTR ANDbits,
LPSTR XORbits);
```

Beschreibung Diese Funktion erzeugt ein Symbol der angegebenen Breite und Höhe mit dem vorgegebenen Bitmuster.

Instance: Bezeichnet eine Instanz des Moduls, welches das Symbol erzeugt.

Width: Bestimmt die Breite des Symbols in Pixeln.

Height: Bestimmt die Höhe des Symbols in Pixeln.

Planes: Bestimmt die Anzahl der Ebenen in der Exclusives-ODER-Maske des Symbols.

BitsPixel: Bestimmt die Anzahl der Bits pro Pixel in der Exklusives-ODER-Maske des Symbols.

ANDbits: Zeigt auf ein Byte-Array, das die Bitwerte für die UND-Maske des Symbols enthält. Dieses Array muß eine monochrome Maske bezeichnen.

XORbits: Zeigt auf ein Byte-Array, das die Bitwerte für die Exklusives-ODER-Maske des Symbols enthält. Dies können die Bits eines monochromen oder geräteabhängigen farbigen Bitmaps sein.

Rückgabewert Der Rückgabewert bezeichnet ein Symbol, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls ist er NULL.

CreateMenu

Syntax `HMENU CreateMenu(void);`

Beschreibung Diese Funktion erzeugt ein Menü. Das Menü ist anfangs leer, kann aber mit Menüoptionen gefüllt werden, indem man die Funktionen *AppendMenu* oder *InsertMenu* verwendet.
Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bezeichnet bei erfolgreicher Ausführung der Funktion ein neu erzeugtes Menü. Er ist NULL, wenn das Menü nicht erzeugt werden kann.

Querverweis *AppendMenu*, *InsertMenu*

CreateMetaFile

Syntax `HANDLE CreateMetaFile(LPSTR FileName);`

Beschreibung Diese Funktion erzeugt einen Metafile-Gerätekontext.

FileName: Zeigt auf einen nullterminierten String, der den Namen des Metafile angibt. Wenn der Parameter *FileName* NULL ist, wird ein Gerätekontext für ein Speicher-Metafile zurückgeliefert.

Rückgabewert Der Rückgabewert bezeichnet einen Metafile-Gerätekontext, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls ist er NULL.

CreatePalette

Syntax `HPALETTE CreatePalette(LPLOGPALETTE LogPalette);`

Beschreibung Diese Funktion erzeugt eine virtuelle Farbpalette.

LogPalette: Zeigt auf eine *LOGPALETTE*-Datenstruktur, die Informationen über die Farben der virtuellen Palette enthält.

Rückgabewert Der Rückgabewert bezeichnet eine virtuelle Palette, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

CreatePatternBrush

Syntax `HBRUSH CreatePatternBrush(HBITMAP Bitmap);`

Beschreibung Diese Funktion erzeugt einen virtuellen Pinsel, der das vom Parameter *Bitmap* angegebene Muster hat. Der Pinsel kann später für jede Geräteeinheit ausgewählt werden, die Rastero-

perationen unterstützt. Weitere Informationen finden sich in der Beschreibung der Funktion *GetDeviceCaps* und in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »DC_XXXX«.

Bitmap: Bezeichnet das Bitmap, vorausgesetzt, daß es durch eine der Funktionen *CreateBitmap*, *CreateBitmapIndirect*, *LoadBitmap* oder *CreateCompatibleBitmap* erzeugt wurde. Die minimale Größe eines Bitmaps, das in einem Füllmuster benutzt werden soll, beträgt 8-auf-8.

Rückgabewert Der Rückgabewert bezeichnet einen virtuellen Pinsel, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Hinweis Ein Musterpinsel kann gelöscht werden, ohne das zugeordnete Bitmap zu beeinflussen, indem man die Funktion *DeleteObject* verwendet. So kann das Bitmap verwendet werden, um eine beliebige Anzahl von Musterpinseln zu erzeugen.

Ein Pinsel, der unter Verwendung eines monochromen (eine Ebene, ein Bit pro Pixel) Bitmap erzeugt wurde, wird mit der aktuellen Text- und Hintergrundfarbe gezeichnet. Pixel, die durch ein auf 0 gesetztes Bit repräsentiert werden, werden in der aktuellen Textfarbe gezeichnet. Pixel, bei denen das Bit auf 1 gesetzt ist, werden in der aktuellen Hintergrundfarbe gezeichnet.

Querverweis *CreateBitmap*, *CreateBitmapIndirect*, *LoadBitmap*, *CreateCompatibleBitmap*, *DeleteObject*, *GetDeviceCaps*

CreatePen

Syntax
```
HPEN CreatePen(int PenStyle, int Width, COLORREF Color);
```

Beschreibung Diese Funktion erzeugt einen virtuellen Stift, der den Stil mit der angegebenen Breite und Farbe hat. Der Stift kann später als aktueller Stift für jede Geräteeinheit ausgewählt werden.

PenStyle: Gibt den Stil des Stifts an. Er kann jeden der folgenden Werte annehmen (vergleiche Kapitel 4 »Windows-Stile und -Konstanten«, Abschnitt »PS_XXXX«):

Stiftstil	Wert	Aussehen
PS_SOLID	0	durchgezogene Linie
PS_DASH	1	gestrichelte Linie
PS_DOT	2	gepunktete Linie
PS_DASHDOT	3	_._._._._._._._
PS_DASHDOTDOT	4	_.._.._.._.._.._
PS_NULL	5	unsichtbar
PS_INSIDEFRAME	6	Innenrahmen von Polygonen und Doppellinien

Wenn die Breite des Stifts größer als 1 und der Stiftstil *PS_IN-*

SIDEFRAME ist, wird die Linie innerhalb des Rahmens aller Grundfunktionen mit Ausnahme von Polygonen und Linienzügen gezeichnet. Der Stift wird mit einer angenäherten Farbe gezeichnet, wenn die Stiftfarbe keinem verfügbaren RGB-Wert entspricht. Der Stil *PS_INSIDEFRAME* ist identisch mit *PS_SOLID*, wenn die Stiftbreite kleiner oder gleich 1 ist.

Width: Bestimmt die Breite des Stifts in virtuellen Einheiten.

Color: Bestimmt die Farbe des Stifts.

Rückgabewert Der Rückgabewert bezeichnet einen virtuellen Stift, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Hinweis Stifte mit einer physikalischen Breite von mehr als einem Pixel haben immer den Stil *PS_NULL* oder *PS_SOLID* oder werden angenähert, wenn der Stiftstil *PS_INSIDEFRAME* ist.

CreatePenIndirect

Syntax
```
HPEN CreatePenIndirect(LOGPEN FAR* LogPen);
```

Beschreibung Diese Funktion erzeugt einen virtuellen Stift, der den Stil, die Breite und Farbe hat, die in der Datenstruktur bestimmt sind, auf die der Parameter *LogPen* zeigt.

LogPen: Zeigt auf eine *LOGPEN*-Datenstruktur, die Informationen über den virtuellen Stift enthält.

Rückgabewert Der Rückgabewert bezeichnet einen virtuellen Stift, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Hinweis Stifte mit einer physikalischen Breite von mehr als einem Pixel haben immer den Stil *PS_NULL* oder *PS_SOLID* oder werden angenähert, wenn der Stiftstil *PS_INSIDEFRAME* ist.

CreatePolygonRgn

Syntax
```
HRGN CreatePolygonRgn(LPPOINT Points, int Count,
int PolyFillMode);
```

Beschreibung Diese Funktion erzeugt eine Polygon-Region.

Points: Zeigt auf ein Array von *POINT*-Datenstrukturen. Jeder Punkt gibt die x- und y-Koordinaten eines Scheitelpunktes des Polygons an.

Count: Bestimmt die Anzahl der Punkte im Array.

PolyFillMode: Bestimmt den Polygon-Füllmodus, der benutzt werden soll, um die Region auszufüllen. Er kann *Alternate* oder *Winding* sein. Die Modi werden in Kapitel 4 »Windows-Stile und -Konstanten« in Abschnitt »Polygon-Füllmodi« beschrieben.

Rückgabewert Der Rückgabewert bezeichnet eine neue Region, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Querverweis *SetPolyFillMode*

CreatePolyPolygonRgn

Syntax
```
HRGN CreatePolyPolygonRgn(LPPOINT Points,
LPINT PolyCounts, int Counts, int PolyFillMode);
```

Beschreibung Diese Funktion erzeugt eine Region, die aus einer Reihe von geschlossenen Polygonen besteht. Die Region wird unter Verwendung des durch den Parameter *PolyFillMode* angegebenen Modus ausgefüllt.

Die Polygone können sich überschneiden, müssen es aber nicht.

Points: Zeigt auf ein Array von *POINT*-Datenstrukturen, die die Scheitelpunkte der Polygone bestimmen. Jedes Polygon muß geschlossen werden, dies wird nicht automatisch durchgeführt. Die Polygone werden nacheinander angegeben.

PolyCounts: Zeigt auf ein Integer-Array. Jeder Integer bestimmt die Anzahl von Punkten eines der Polygone des *Points*-Array.

Count: Bestimmt die Gesamtanzahl der Integer im Array *PolyCounts*.

PolyFillMode: Bestimmt den Füllmodus für die Region. Der Parameter kann einen der folgenden Werte annehmen:

Wert	Bedeutung
Alternate	Füllt alternierende Flächen.
Winding	Füllt jede Fläche.

Rückgabewert Der Rückgabewert bezeichnet die Region, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls ist er NULL.

Hinweis Generell unterscheiden sich die Polygon-Füllmodi nur in Fällen, wo ein komplexes, überlappendes Polygon gefüllt werden muß (zum Beispiel ein fünfseitiges Polygon, das einen fünfzackigen Stern mit einem Fünfeck in der Mitte bildet). In solchen Fällen füllt der *Alternate*-Modus jede eingeschlossene Region innerhalb des Polygons (d.h. die Punkte des Sterns), der *Winding*-Modus jedoch füllt alle Regionen (d.h. die Punkte und das Fünfeck).

Wenn der Füllmodus *Alternate* ist, füllt die GDI den Bereich zwischen geraden und ungeraden Polygonseiten auf jeder Scan-Zeile. Das heißt, daß die GDI den Bereich zwischen der ersten und zweiten, der dritten und vierten Seite füllt, usw.

Um alle Teile einer Region zu füllen, veranlaßt der *Winding*-

Modus die GDI, eine Umrandung zu berechnen und zu zeichnen, die die Region einschließt, aber nicht überlappt.

Zum Beispiel wird im *Winding*-Modus das fünfseitige Polygon, das den Stern bildet, als ein zehnseitiges Polygon ohne sich überschneidende Seiten berechnet; der daraus resultierende Stern wird ausgefüllt.

CreatePopupMenu

Syntax `HMENU CreatePopupMenu(void);`

Beschreibung Diese Funktion erzeugt leeres Pop-up-Menü und gibt dessen Handle zurück. Eine Anwendung fügt Einträge zu einem Pop-up-Menü hinzu, indem sie die Funktionen *InsertMenu* und *AppendMenu* aufruft.

Eine Anwendung kann das Pop-up-Menü zu einem bestehenden Menü oder Pop-up-Menü hinzufügen, oder sie kann eine Auswahl im Pop-up-Menü anzeigen und verfolgen, indem sie die Funktion *TrackPopupMenu* aufruft.

Diese Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bezeichnet das neuerzeugte Menü. Er ist NULL, wenn das Menü nicht erzeugt werden kann.

Querverweis *AppendMenu, InsertMenu, TrackPopupMenu*

CreateRectRgn

Syntax `HRGN CreateRectRgn(int X1, int Y1, int X2, int Y2);`

Beschreibung Diese Funktion erzeugt eine rechteckige Region.

X1, Y1: Bestimmt die x- bzw. y-Koordinate der oberen linken Ecke der Region.

X2, Y2: Bestimmt die x- bzw. y-Koordinate der unteren rechten Ecke der Region.

Rückgabewert Der Rückgabewert bezeichnet eine neue Region, wenn die Funktion erfolgreich ausgeführt werden konnte, andernfalls ist er NULL.

Hinweis Die Breite des Rechtecks, die durch den absoluten Wert der Differenz *X2 - X1* gegeben ist, darf 32767 Einheiten nicht überschreiten. Diese Beschränkung gilt auch für die Höhe des Rechtecks (*Y2 - Y1*).

CreateRectRgnIndirect

Syntax `HRGN CreateRectRgnIndirect(LPRECT Rect);`

Beschreibung Diese Funktion erzeugt eine rechteckige Region.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die Koordinaten der oberen linken und unteren rechten Ecke der Region enthält.

Rückgabewert Der Rückgabewert bezeichnet eine neue Region, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Hinweis Die Breite des Rechtecks darf 32767 Einheiten nicht überschreiten. Diese Beschränkung gilt auch für die Höhe des Rechtecks.

CreateRoundRectRgn

Syntax
```
HRGN CreateRoundRectRgn(int X1, int Y1, int X2, int Y2,
int X3, int Y3);
```

Beschreibung Diese Funktion erzeugt eine rechteckige Region mit abgerundeten Ecken.

X1, Y1: Bestimmt die x- bzw. y-Koordinate der oberen linken Ecke der Region.

X2, Y2: Bestimmt die x- bzw. y-Koordinate der unteren rechten Ecke der Region.

X3, Y3: Bestimmt die Breite bzw. die Höhe der Ellipse, die verwendet wird, um die abgerundeten Ecken zu erzeugen.

Rückgabewert Der Rückgabewert bezeichnet eine neue Region, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

Hinweis Die Breite des Rechtecks, die durch den absoluten Wert der Differenz *X2 - X1* gegeben ist, darf 32767 Einheiten nicht überschreiten. Diese Beschränkung gilt auch für die Höhe des Rechtecks *(Y2 - Y1)*.

CreateSolidBrush

Syntax
```
HBRUSH CreateSolidBrush(COLORREF Color);
```

Beschreibung Diese Funktion erzeugt einen virtuellen Pinsel mit der angegebenen Farbe. Der Pinsel kann später für jede Geräteeinheit als aktueller Pinsel gewählt werden.

Color: 32-Bit-Wert vom Typ COLORREF, der die Farbe des Pinsels bestimmt.

Rückgabewert Der Rückgabewert bezeichnet einen virtuellen Pinsel, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten ist er NULL.

CreateWindow

Syntax
```
HWND CreateWindow(LPSTR ClassName, LPSTR WindowName,
DWORD Style, int X, int Y, int Width, int Height,
```

```
HWND WndParent, HMENU Menu, HANDLE Instance,
LPSTR Param);
```

Beschreibung Diese Funktion erzeugt ein überlappendes, ein Pop-up-, oder ein untergeordnetes Fenster. Sie bestimmt die Fensterklasse, die Fensterüberschrift, den Fensterstil und (optional) die anfängliche Position und Größe des Fensters. *CreateWindow* bestimmt auch das übergeordnete Fenster (falls vorhanden) und das Menü des Fensters.

Für überlappende, Pop-up- und untergeordnete Fenster übergibt *CreateWindow* die Botschaften *WM_CREATE*, *WM_GETMINMAXINFO* und *WM_NCCREATE* an das Fenster. Der Parameter *lParam* der Botschaft *WM_CREATE* enthält einen Zeiger auf eine *CREATESTRUCT*-Datenstruktur. Wenn der Stil *WS_VISIBLE* gegeben ist, sendet *CreateWindow* alle Botschaften an das Fenster, die erforderlich sind, um das Fenster zu aktivieren und zu zeigen.

Bestimmt der Fensterstil eine Titelleiste, wird die Fensterüberschrift, auf die der Parameter *WindowName* zeigt, in der Titelleiste angezeigt. Wird *CreateWindow* verwendet, um Steuerelemente wie Aktionsschalter, Auswahlfelder und Editierfelder zu erzeugen, zeigt der Parameter *WindowName* auf den Text der Steuerelemente.

ClassName: Zeigt auf einen nullterminierten String, der die Fensterklasse benennt. Der Klassenname kann jeder Name sein, der durch die Funktion *RegisterClass* eingetragen wird, oder der Name einer der vordefinierten Steuerelementeklassen, die im folgenden »Hinweis« beschrieben werden.

WindowName: Zeigt auf einen nullterminierten String, der den Fensternamen enthält.

Style: Bestimmt den Stil des Fensters, das erzeugt werden soll. Es kann jede Kombination der Stile verwendet werden, die in Kapitel 4 »Windows-Stile und -Konstanten« in den Abschnitten »DS_XXXX«, »WS_XXXX«, »BS_XXXX«, »CBS_XXXX«, »ES_XXXX«, »LBS_XXXX«, »SBS_XXXX« und »SS_XXXX« beschrieben sind.

X: Bestimmt die anfängliche x-Position des Fensters. Bei einem überlappenden oder Pop-up-Fenster ist der Parameter *X* die ursprüngliche x-Koordinate der oberen linken Ecke des Fensters in Bildschirmkoordinaten. Wenn dieser Wert *CW_USEDEFAULT* ist, wählt Windows die voreingestellte Position für die obere linke Ecke des Fensters. Bei einem untergeordneten Fenster ist *X* die x-Koordinate der oberen linken Ecke des Fensters im Client-Bereich seines übergeordneten Fensters.

Y: Gibt die anfängliche y-Position des Fensters an. Bei einem überlappenden Fenster ist *Y* die ursprüngliche y-Koordinate der oberen linken Ecke des Fensters. Bei einem Pop-up Fenster

ist *Y* die y-Koordinate der oberen linken Ecke des Pop-up Fensters in Bildschirmkoordinaten. Bei Listen ist *Y* die y-Koordinate der oberen linken Ecke des Client-Bereichs des Steuerelements. Bei einem untergeordneten Fenster ist *Y* die y-Koordinate der oberen linken Ecke des untergeordneten Fensters. Alle diese Koordinaten beziehen sich auf das Fenster selbst, nicht auf den Client-Bereich des Fensters.

Width: Gibt die Breite des Fensters in Geräteeinheiten an. Bei überlappenden Fenstern ist der Parameter *Width* entweder die Breite des Fensters in Bildschirmkoordinaten oder *CW_USEDEFAULT*. Wenn *Width CW_USEDEFAULT* ist, wählt Windows eine voreingestellte Breite und Höhe für das Fenster. Die voreingestellte Breite erstreckt sich von der anfänglichen x-Position bis zum rechten Rand des Bildschirms, und die voreingestellte Höhe erstreckt sich von der anfänglichen y-Position bis zum oberen Ende des Symbolbereichs.

Height: Gibt die Höhe des Fensters in Geräteeinheiten an. Bei überlappenden Fenstern ist der Parameter *Height* die Höhe des Fensters in Bildschirmkoordinaten. Wenn der Parameter *Width CW_USEDEFAULT* ist, ignoriert Windows *Height*.

WndParent: Bezeichnet das übergeordnete oder das besitzende Fenster des Fensters, welches erzeugt wurde. Wenn man ein untergeordnetes oder ein zugehöriges Fenster erzeugt, muß ein gültiges Fenster-Handle angegeben werden. Ein zugehöriges Fenster ist ein überlapptes Fenster, das entfernt wird, wenn sein besitzendes Fenster entfernt oder unsichtbar gemacht oder wenn sein Besitzer als Symbol dargestellt wird. Ein zugehöriges Fenster wird immer auf der Oberfläche seines besitzenden Fensters angezeigt. Bei Pop-up-Fenstern kann ein Handle angegeben werden, dies ist aber nicht erforderlich. Wenn das Fenster kein übergeordnetes Fenster hat oder es keinem anderen Fenster zugehörig ist, muß der Parameter *WndParent* auf NULL gesetzt werden.

Menu: Bezeichnet ein Menü oder enthält den Bezeichner eines untergeordneten Fensters. Die Bedeutung hängt vom Fensterstil ab. Bei überlappenden oder Pop-up-Fenstern bezeichnet der Parameter *Menu* das Menü, welches mit dem Fenster benutzt werden soll. Er kann NULL sein, wenn das Klassenmenü verwendet werden soll. Bei untergeordneten Fenstern gibt *Menu* den Bezeichner eines untergeordneten Fensters an: einen Integer, der von einem Dialogfenster benutzt wird, um seinem übergeordneten Fenster Ereignisse mitzuteilen (wie beispielsweise die Botschaft *EN_HSCROLL*). Der Bezeichner des untergeordneten Fensters wird von der Anwendung ermittelt und sollte für alle untergeordneten Fenster mit dem gleichen übergeordneten Fenster eindeutig sein.

Instance: Bezeichnet die Instanz des Moduls, das dem Fenster zugeordnet werden soll.

Param: Zeigt auf einen Wert, der dem Fenster durch die *CREATESTRUCT*-Datenstruktur übergeben wird, auf die der Parameter *lParam* der Botschaft *WM_CREATE* zeigt. Wenn eine Anwendung *CreateWindow* aufruft, um ein Client-Fenster zu erzeugen, muß *lParam* auf eine *CLIENTCREATESTRUCT*-Datenstruktur zeigen.

Rückgabewert Der Rückgabewert bezeichnet das neue Fenster. Er ist NULL, wenn das Fenster nicht erzeugt wurde.

Hinweis Bei überlappten Fenstern, für die der Parameter *X* auf *CW_USEDEFAULT* gesetzt ist, kann der Parameter *Y* einer der bei der Funktion *ShowWindow* beschriebenen Parameter für Darstellungsstile sein oder er kann für das erste überlappte Fenster, das von der Anwendung erzeugt werden soll, der der Funktion *WinMain* übergebene Parameter *nCmdShow* sein.

Die folgende Übersicht beschreibt die Klassen der Steuerelemente, die in Windows vordefiniert sind.

Klasse	Bedeutung
BUTTON	Bestimmt ein kleines rechteckiges untergeordnetes Fenster, das einen Aktionsschalter darstellt, den der Anwender durch Anklicken ein- oder ausschalten kann. Schalter können allein oder in Gruppen verwendet werden. Sie können beschriftet sein oder ohne Text erscheinen. Schalter ändern üblicherweise ihr Aussehen, wenn der Anwender sie anklickt.
COMBOBOX	Bestimmt ein Steuerelement, das aus einem Auswahlfeld mit einem Editierfeld und einer Liste besteht. Die Liste kann zu jeder Zeit oder dann angezeigt werden, wenn der Anwender ein Feld neben dem Auswahlfeld anwählt bzw. anklickt. Abhängig vom Stil des Kombinationfensters kann der Anwender den Inhalt des Auswahlfeldes editieren oder nicht. Wenn die Liste sichtbar ist, verursacht das Eingeben von Zeichen im Auswahlfenster, daß der erste Listeneintrag, der zu den eingegebenen Zeichen paßt, hervorgehoben dargestellt wird. Umgekehrt bewirkt das Anwählen eines Eintrags in der Liste, daß der ausgewählte Text im Auswahlfeld angezeigt wird.

Klasse	Bedeutung
EDIT	Bestimmt ein rechteckiges untergeordnetes Fenster, in dem der Anwender mit der Tastatur Text eingeben kann. Der Anwender wählt das Steuerelement und

	gibt ihm den Fokus, indem er es anklickt oder sich durch Verwendung der Tab-Taste zu ihm hinbewegt. Der Anwender kann Text eingeben, wenn das Steuerelement ein blinkendes Caret anzeigt. Die Maus kann verwendet werden, um den Mauszeiger zu bewegen und Zeichen auszuwählen, die ersetzt werden sollen oder um den Mauszeiger zum Einfügen von Zeichen zu positionieren. Backspace dient zum Löschen von Zeichen. Editierfelder verwenden die Systemschrift mit variablem Zeichenabstand und zeigen ANSI-Zeichen an. Anwendungen, die compiliert wurden, um unter früheren Versionen von Windows zu laufen, zeigen Text mit einer Systemschrift und festen Zeichenabständen an, wenn sie nicht durch das Windows 3.0 Dienstprogramm MARK mit der Option MEMORY FONT gekennzeichnet worden sind. Eine Anwendung kann die Botschaft *WM_SETFONT* an das Editierfeld senden, um die voreingestellte Schrift zu wechseln. Editierfelder expandieren Tabulatorzeichen zu so vielen Leerzeichen, wie nötig sind, um den Mauszeiger zum nächsten Tabulatorstop zu bewegen. Windows geht von einem Abstand von acht Zeichen zwischen den Tabulatorstops aus.
LISTBOX	*Be*stimmt eine Liste von Strings. Dieses Steuerelement wird immer dann verwendet, wenn eine Anwendung eine Liste von Namen anzeigen soll, die der Anwender anschauen und auswählen kann. Der Anwender kann einen String auswählen, indem er auf ihn zeigt und klickt. Wenn ein String gewählt ist, wird er hervorgehoben dargestellt und eine Botschaft an das übergeordnete Fenster übergeben. Eine horizontale oder vertikale Bildlaufleiste kann verwendet werden, um Listen zu rollen, die zu lang für das Fenster sind. Die Liste macht die Bildlaufleisten automatisch sichtbar, wenn dies notwendig ist.
MDICLIENT	Bestimmt ein MDI-Client-Fenster. Das MDI-Client-Fenster empfängt Botschaften, die die untergeordneten Fenster der MDI-Anwendung steuern. Die empfohlenen Stilbits sind *WS_CLIPCHILDREN* und *WS_CHILD*. Um ein bildlauffähiges MDI-Client-Fenster zu erzeugen, das es dem Anwender erlaubt, Bildlauf für untergeordnete MDI-Fenster durchzuführen, kann eine Anwendung auch die Stile *WS_HSCROLL* und *WS_VSCROLL* verwenden.

Klasse	Bedeutung
SCROLLBAR	Bestimmt ein Rechteck, das einen Positionszeiger enthält und Richtungspfeile an beiden Enden hat.

	Die Bildlaufleiste schickt immer dann eine Benachrichtigungsbotschaft an ihr übergeordnetes Fenster, wenn der Anwender das Steuerelement anklickt. Das übergeordnete Fenster ist, wenn nötig, für das Aktualisieren der Position des Positionszeigers verantwortlich. Bildlaufleistenelemente haben das gleiche Aussehen und die gleiche Funktion wie Bildlaufleisten, die in gewöhnlichen Fenstern verwendet werden. Im Gegensatz zu Bildlaufleisten können Bildlaufleistenelemente überall in einem Fenster positioniert werden, wann immer sie benötigt werden, um bildlauffähige Eingaben für ein Fenster bereitzustellen. Die Klasse *SCROLLBAR* schließt auch ein Größenveränderungssymbol ein. Ein Größenveränderungssymbol ist ein kleines Rechteck, das der Anwender verwenden kann, um die Größe eines Fensters zu ändern.
STATIC	Bestimmt ein einfaches Textfeld, Kästchen oder Rechteck, das verwendet werden kann, um andere Steuerelemente zu beschriften, einzurahmen oder abzutrennen. Statische Steuerelemente nehmen keine Eingaben entgegen und liefern keine Ausgaben.

Querverweis *RegisterClass, WM_CREATE, WM_GETMINMAXINFO, WM_NCCREATE*

CreateWindowEx

Syntax

```
HWND CreateWindowEx(DWORD ExStyle, LPSTR ClassName,
LPSTR WindowName, DWORD Style, int X, int Y,
int Width, int Height, HWND WndParent, HMENU Menu,
HANDLE Instance, LPSTR Param);
```

Beschreibung Diese Funktion erzeugt ein überlapptes, ein Pop-up- oder ein untergeordnetes Fenster mit einem im Parameter *ExStyle* angegebenen erweiterten Stil. Im übrigen ist diese Funktion identisch mit der Funktion *CreateWindow*. In der Beschreibung dieser Funktion finden sich ausführliche Informationen über die Erzeugung eines Fensters.

ExStyle: Bestimmt den erweiterten Stil des Fensters, das erzeugt wird.

ClassName: Zeigt auf einen nullterminierten String, der den Namen der Fensterklasse enthält.

WindowName: Zeigt auf einen nullterminierten String, der den Fensternamen enthält.

Style: Bestimmt den Stil des Fensters, das erzeugt wird.

X: Bestimmt die anfängliche x-Position des Fensters.

Y: Bestimmt die anfängliche y-Position des Fensters.

Width: Bestimmt die Breite des Fensters in Geräteeinheiten.

Height: Bestimmt die Höhe des Fensters in Geräteeinheiten.

WndParent: Bezeichnet das übergeordnete oder das besitzende Fenster des Fensters, das erzeugt wird.

Menu: Bestimmt den Bezeichner eines Menüs oder eines untergeordneten Fensters. Die Bedeutung ist vom Fensterstil abhängig.

Instance: Bezeichnet die Instanz des Moduls, das dem Fenster zugeordnet werden soll.

Param: Zeigt auf den Wert, der dem Fenster in der *CREATESTRUCT*-Datenstruktur übergeben wird, auf die der Parameter *lParam* der Botschaft *WM_CREATE* zeigt.

Rückgabewert Der Rückgabewert bezeichnet das neue Fenster. Er ist NULL, wenn das Fenster nicht erzeugt wird.

Hinweis Eine Beschreibung der erweiterten Fensterstile finden sich im Abschnitt »WS_EX_XXXX« in Kapitel 4 »Windows-Stile und -Konstanten«. Die normalen Fensterstile sind ebenfalls in Kapitel 4 in den Abschnitten »DS_XXXX«, »WS_XXXX«, »BS_XXXX«, »CBS_XXXX«, »ES_XXXX«, »LBS_XXXX«, »SBS_XXXX« und »SS_XXXX« beschrieben.

Querverweis *CreateWindow, WM_PARENTNOTIFY*

DebugBreak

Syntax
```
void DebugBreak(void);
```

Beschreibung Diese Funktion erzwingt eine Unterbrechung zum Debugger. Sie hat keine Parameter.

Rückgabewert Keiner.

DefDlgProc

Syntax
```
LONG DefDlgProc(HWND Dlg, WORD Msg, WORD wParam,
LONG lParam);
```

Beschreibung Diese Funktion ermöglicht die Bearbeitung aller Windows-Botschaften, die von Dialogfenstern individueller Klassen nicht abgefangen werden. Alle Fensterbotschaften, die nicht explizit von der Fensterfunktion bearbeitet werden, müssen an *DefDlgProc*, nicht an *DefWindowProc* weitergegeben werden. Dies stellt sicher, daß Botschaften, die vom zugehörigen Fen-

ster nicht bearbeitet werden können, trotzdem ordnungsgemäß abgearbeitet werden.

Dlg: Bezeichnet das Dialogfenster.

Msg: Bestimmt die Nummer der Botschaft.

wParam: Bestimmt 16 Bits zusätzlicher, botschaftsabhängiger Informationen.

lParam: Bestimmt 32 Bits zusätzlicher, botschaftsabhängiger Informationen.

Rückgabewert Der Rückgabewert hängt von der übergebenen Botschaft ab. Er bestimmt das Ergebnis der bearbeiteten Botschaft.

Hinweis Anwendungen können Dialogfenster durch den Aufruf einer der folgenden Funktionen erstellen:

Funktion	Beschreibung
CreateDialog	Erzeugt ein nicht-modales Dialogfenster.
CreateDialogIndirect	Erzeugt ein nicht-modales Dialogfenster.
CreateDialogIndirectParam	Erzeugt ein nicht-modales Dialogfenster. und übergibt Daten an dieses Fenster.
CreateDialogParam	Erzeugt ein nicht-modales Dialogfenster und übergibt Daten an dieses Fenster.
DialogBox	Erzeugt ein modales Dialogfenster.
DialogBoxIndirect	Erzeugt ein modales Dialogfenster.
DialogBoxIndirectParam	Erzeugt ein modales Dialogfenster und übergibt Daten an dieses Fenster.
DialogBoxParam	Erzeugt ein modales Dialogfenster und übergibt Daten an dieses Fenster.

DeferWindowPos

Syntax

```
HANDLE DeferWindowPos(HANDLE WinPosInfo, HWND Wnd,
HWND WndInsertAfter, int X, int Y, int cX, int cY,
WORD Flags);
```

Beschreibung Diese Funktion aktualisiert die durch den Parameter *WinPosInfo* bezeichnete Datenstruktur für mehrere Fensterpositionen hinsichtlich des in *Wnd* angegebenen Fensters. Sie gibt das Handle der aktualisierten Struktur zurück. Die Funktion *EndDeferWindowPos* verwendet die Informationen in dieser Struktur, um Position und Größe mehrerer Fenster gleichzeitig zu verändern. Die Struktur wird von der Funktion *BeginDeferWindowPos* erzeugt.

Die Parameter *X* und *Y* bestimmen die neue Position, *cX* und *cY* die neue Größe des Fensters.

WinPosInfo: Bezeichnet eine Datenstruktur für mehrere Fen-

sterpositionen, die Informationen über Größe und Position eines oder mehrerer Fenster enthält.

Die Struktur wird von der Funktion *BeginDeferWindowPos* oder vom letzten Aufruf der Funktion *DeferWindowPos* zurückgegeben.

Wnd: Bezeichnet das Fenster, dessen aktualisierte Daten in der Struktur abgelegt werden.

WndInsertAfter: Bezeichnet das Fenster, das dem zu aktualisierenden vorhergeht.

X: Bestimmt die x-Koordinate der oberen linken Ecke des Fensters.

Y: Bestimmt die y-Koordinate der oberen linken Ecke des Fensters.

cX: Bestimmt die neue Breite des Fensters.

cY: Bestimmt die neue Höhe des Fensters.

Flags: Bestimmt einen von acht möglichen 16-Bit-Werten, die Größe und Position des Fensters beeinflussen. Möglich sind folgende Werte:

Wert	Bedeutung
SWP_DRAWFRAME	Zeichnet den in der Fensterklassendefinition definierten Rahmen um das Fenster.
SWP_HIDEWINDOW	Versteckt das Fenster.
SWP_NOACTIVATE	Fenster wird nicht aktiviert.
SWP_NOMOVE	Erhält die aktuelle Position, ignoriert die Parameter *X* und *Y*.
SWP_NOREDRAW	Änderungen werden nicht dargestellt.
SWP_NOSIZE	Erhält die aktuelle Größe, ignoriert die Parameter *cX* und *cY*.
SWP_NOZORDER	Erhält die aktuelle Ordnung, ignoriert den Parameter *WndInsertAfter*.
SWP_SHOWWINDOW	Stellt das Fenster dar.

Rückgabewert

Der Rückgabewert bezeichnet die aktualisierte Datenstruktur für mehrere Fensterpositionen. Das zurückgegebene Handle kann von dem im Parameter *WinPosInfo* übergebenen abweichen. Das neue Handle sollte beim nächsten Aufruf an die Funktionen *DeferWindowPos* oder *EndDeferWindowPos* übergeben werden.

Der Rückgabewert ist dann NULL, wenn die zur Verfügung stehenden Systemressourcen für die vollständige Bearbeitung dieser Funktion nicht ausreichen.

Hinweis

Ist das Flag *SWP_NOZORDER* nicht gesetzt, positioniert Windows das durch *Wnd* bezeichnete Fenster nach dem durch den

Parameter *WndInsertAfter* festgelegte Fenster. Ist *WndInsertAfter* NULL, wird das durch *Wnd* bezeichnete Fenster an den Anfang der Liste, ist der Parameter *1*, wird es an das Ende der Liste gesetzt.

Ist eines der Flags *SWP_SHOWWINDOW* oder *SWP_HIDEWINDOW* gesetzt, können Bildlauf und Verschiebungen nicht gleichzeitig durchgeführt werden.

Die Koordinaten untergeordneter Fenster beziehen sich auf die linke obere Ecke des Client-Bereichs des übergeordneten Fensters.

Querverweis *BeginDeferWindowPos, EndDeferWindowPos*

DefFrameProc

Syntax
```
LONG DefFrameProc(HWND Wnd, HWND MDIClient, WORD Msg,
WORD wParam, LONG lParam);
```

Beschreibung Diese Funktion unterstützt die normale Bearbeitung von Windows-Botschaften, die von der Fensterfunktion eines MDI-Rahmenfensters nicht bearbeitet werden. Sämtliche Fensterbotschaften, die nicht explizit von der Fensterfunktion abgearbeitet werden, müssen an *DefFrameProc*, nicht an *DefWindowProc* weitergegeben werden.

Wnd: Bezeichnet das MDI-Rahmenfenster.

MDIClient: Bezeichnet das MDI-Client-Fenster.

Msg: Bestimmt die Nummer der Botschaft.

wParam: Bestimmt 16 Bits zusätzlicher, botschaftsabhängiger Informationen.

lParam: Bestimmt 32 Bits zusätzlicher, botschaftsabhängiger Informationen.

Rückgabewert Der Rückgabewert hängt von der übergebenen Botschaft ab. Er bestimmt das Ergebnis der bearbeiteten Botschaft. Ist der Parameter *MDICClient* NULL, entspricht der Rückgabewert dem der Funktion *DefWindowProc*.

Hinweis Bearbeitet die Fensterprozedur einer Anwendung eine Botschaft nicht, so wird sie normalerweise an die Funktion *DefWindowProc* übergeben. MDI-Anwendungen verwenden zur voreingestellten Bearbeitung von Botschaften die Funktionen *DefFrameProc* und *defmdiChildProc*.

Botschaften, die normalerweise an *DefWindowProc* weitergegeben würden, sollten stattdessen an *DefFrameProc* weitergegeben werden. Dies gilt für allgemeine Botschaften und für *WM_SETTEXT*. Zudem können folgende Botschaften von *DefFrameProc* bearbeitet werden:

Botschaft	Standardbearbeitung von *DefFrameProc*
WM_COMMAND	Das Rahmenfenster einer MDI-Anwendung erhält die Botschaft *WM_COMMAND* zur Aktivierung eines bestimmten untergeordneten MDI-Fensters Die gleichzeitig übergebene Fenster-ID ist der Bezeichner des durch Windows festgelegten untergeordneten MDI-Fensters. Begonnen wird mit dem ersten Bezeichner, der von einer Anwendung bei Erstellung des untergeordneten MDI-Fensters bestimmt wurde – der Wert darf sich nicht mit den Bezeichnern der Menüeinträge überschneiden.
WM_MENUCHAR	Wird die Tastenkombination Alt und Bindestrich - gedrückt, wird das Systemmenü des aktiven untergeordneten MDI-Fensters angewählt.
WM_NEXTMENU	Diese Botschaft veranlaßt die Auswahl des Systemmenüs des aktiven untergeordneten MDI-Fensters.
WM_SETFOCUS	*DefFrameProc* übergibt den Fokus an den MDI-Client, der ihn an das aktive untergeordnete MDI-Fenster weiterleitet.
WM_SIZE	Wenn die Prozedur des Rahmenfensters diese Botschaft an *DefFrameProc* weitergibt, so wird die Größe des untergeordneten MDI-Fensters an den neuen Client-Bereich angepaßt. Bringt die Prozedur des Rahmenfensters den MDI-Client auf eine abweichende Größe, sollte sie die Botschaft nicht an die Funktion *DefWindowProc* weitergeben.

DefHookProc

Syntax

```
DWORD DefHookProc(int Code, WORD wParam, DWORD lParam,
FARPROC FAR * NextHook);
```

Beschreibung

Diese Funktion ruft die folgende Funktion in einer Reihe von Filterverwaltungsfunktionen auf. Eine Filterverwaltungsfunktion bearbeitet Ereignisse, bevor sie an die botschaftsbearbeitende Schleife einer Anwendung in der Funktion *WinMain* übergeben wird.

Definiert eine Anwendung mehr als eine Filterverwaltungsfunktion mit Hilfe der Funktion *SetWindowsHook*, erstellt Windows eine verkettete Liste bzw. Filterverwaltungskette. Alle

Funktionen desselben Typs werden jeweils in einer Kette positioniert.

Code: Bestimmt den Code der Windows-Filterverwaltungsfunktion (auch Botschaftsfilterfunktion). Diese bestimmt die Art der Bearbeitung der Botschaft.

wParam: Bestimmt den WORD-Parameter der Botschaft, die von der Filterverwaltungsfunktion bearbeitet wird.

lParam: Bestimmt den Longint-Parameter der Botschaft, die von der Filterverwaltungsfunktion bearbeitet wird.

NextHook: Zeigt auf einen Speicherbereich, der die von der Funktion *SetWindowsHook* zurückgegebene Struktur *FARPROC* enthält. Windows ändert den Wert dieses Bereichs nach Aufruf der Funktion *UnhookWindowsHook*.

Rückgabewert Der Rückgabewert bestimmt einen Wert, der im direkten Verhältnis zum Parameter *Code* steht.

Querverweis *SetWindowsHook, UnhookWindowsHook*

DefMDIChildProc

Syntax
```
LONG DefMDIChildProc(HWND Wnd, WORD Msg, WORD wParam,
LONG lParam);
```

Beschreibung Diese Funktion unterstützt die normale Bearbeitung von Windows-Botschaften, die von der Fensterfunktion eines untergeordneten MDI-Fensters nicht bearbeitet werden. Alle Fensterbotschaften, die nicht explizit von der Fensterfunktion abgearbeitet werden, müssen an die Funktion *DefMDIChildProc*, nicht an *DefWindowProc* weitergegeben werden.

Wnd: Bezeichnet das untergeordnete MDI-Fenster.

Msg: Bestimmt die Nummer der Botschaft.

wParam: Bestimmt 16 Bits zusätzlicher, botschaftsabhängiger Informationen.

lParam: Bestimmt 32 Bits zusätzlicher, botschaftsabhängiger Informationen.

Rückgabewert Der Rückgabewert hängt von der übergebenen Botschaft ab. Er bestimmt das Ergebnis der bearbeiteten Botschaft.

Hinweis Die Funktion unterstellt, daß das übergeordnete Fenster des durch *Wnd* bezeichneten Fensters mit der Klasse *MDICLIENT* erstellt wurde. Bearbeitet die Fensterprozedur einer Anwendung eine Botschaft nicht, so wird sie normalerweise an die Funktion *DefWindowProc* übergeben, die sie bearbeitet. MDI-Anwendungen verwenden zur Standardbearbeitung von Botschaften dagegen die Funktionen *DefFrameProc* und

DefMDIChildProc. Botschaften, die normalerweise an die Funktion *DefWindowProc* weitergegeben würden, sollten stattdessen an *DefMDIChildProc* weitergegeben werden. Dies gilt für Botschaften aus dem Nicht-Client-Bereich und *WM_SETTEXT*. Zudem werden folgende Botschaften von *DefMDIChildProc* bearbeitet:

Botschaft	Standardbearbeitung von *DefMDIChildProc*
WM_CHILDACTIVATE	Führt Aktivierungsprozeduren durch, wenn untergeordnete Fenster bewegt, dargestellt oder in der Größe verändert werden.
WM_GETMINMAXINFO	Berechnet die Größe eines vergrößerten untergeordneten MDI-Fensters auf Grundlage der aktuellen Größe des MDI-Client-Fensters.
WM_MENUCHAR	Übergibt die Tastenaktivierung an das Rahmenfenster.
WM_MOVE	Berechnet die Bildlaufleisten des MDI-Client-Bereichs neu, wenn diese existieren.
WM_NEXTMENU	Springt zur Menüleiste oder zum Systemmenü des Rahmens zurück.
WM_SETFOCUS	Aktiviert das untergeordnete Fenster, wenn es nicht das aktive MDI-Fenster ist.
WM_SIZE	Führt notwendige Operationen nach Veränderung der Größe (insbesondere Vergrößerung und Wiederherstellung) eines untergeordneten MDI-Fensters durch. Wird die Botschaft nicht an *DefMDIChildProc* übergeben, kann es zu unerwünschten Resultaten kommen.
WM_SYSCOMMAND	Bearbeitet auch den Befehl »Zum nächsten Fenster«.

DefWindowProc

Syntax

```
LONG DefWindowProc(HWND Wnd, WORD Msg, WORD wParam,
LONG lParam);
```

Beschreibung Diese Funktion unterstützt die Standardbearbeitung von Windows-Botschaften, die von der gegebenen Anwendung nicht bearbeitet werden. Alle Fensterbotschaften, die nicht explizit von der Klassenfensterfunktion abgearbeitet werden, müssen an *DefWindowProc* weitergegeben werden.

Wnd: Bezeichnet das Fenster, welches die Botschaft weitergibt.

Msg: Bestimmt die Nummer der Botschaft.

wParam: Bestimmt 16 Bits zusätzlicher, botschaftsabhängiger Informationen.

lParam: Bestimmt 32 Bits zusätzlicher, botschaftsabhängiger Informationen.

Rückgabewert Der Rückgabewert hängt von der übergebenen Botschaft ab. Er zeigt das Ergebnis der bearbeiteten Botschaft an.

DeleteAtom

Syntax
```
ATOM DeleteAtom(ATOM AnAtom);
```

Beschreibung Diese Funktion löscht ein Atom und – wenn der Referenzzähler des Atoms 0 ist – den zugehörigen String aus der Atom-Tabelle.

Der Referenzzähler eines Atoms bestimmt, wie oft ein Atom zur Atom-Tabelle hinzugefügt wurde. Die Funktion *AddAtom* erhöht den Zähler bei jedem Aufruf, die Funktion *DeleteAtom* vermindert den Zähler entsprechend. Der String wird von *DeleteAtom* nur dann gelöscht, wenn der Referenzzähler 0 ist.

AnAtom: Bezeichnet das zu löschende Atom und den zugehörigen nullterminierten String.

Rückgabewert Nach erfolgreicher Ausführung der Funktion wird NULL zurückgeliefert, ansonsten ist der Rückgabewert gleich dem Parameter *AnAtom*. In diesem Fall wird das Atom nicht gelöscht.

DeleteDC

Syntax
```
BOOL DeleteDC(HDC DC);
```

Beschreibung Diese Funktion löscht den angegebenen Gerätekontext. Entspricht der Parameter *DC* dem letzten Gerätekontext des gegebenen Geräts, wird das Gerät entsprechend benachrichtigt.

Alle Speicher- und Systemressourcen des Geräts werden freigegeben.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der Gerätekontext erfolgreich gelöscht wurde, unabhängig davon, ob es sich um den letzten Kontext des Geräts handelt. Tritt ein Fehler auf, ist der Rückgabewert Null.

Hinweis Um einen Gerätekontext zu löschen, dessen Handle die Anwendung durch einen Aufruf der Funktion *GetDC* erhalten hat, muß die Funktion *ReleaseDC* aufgerufen werden.

DeleteMenu

Syntax
```
BOOL DeleteMenu(HMENU Menu, WORD Position, WORD Flags);
```

Beschreibung Diese Funktion löscht einen Eintrag aus dem durch *Menu* bezeichneten Menü. Gehört ein Pop-up-Menü zu diesem Eintrag, entfernt *DeleteMenu* auch das zugehörige Handle, und der vom Pop-up-Menü verwendete Speicher wird freigegeben.

Menu: Bezeichnet das zu ändernde Menü.

Position: Bestimmt den zu löschenden Menüeintrag. Ist *Flags* auf *MF_BYPOSITION* gesetzt, bestimmt *Position* die Position des Menüeintrags. Der erste Menüeintrag hat die Position 0. Ist *Flags* auf *MF_BYCOMMAND* gesetzt, bestimmt *Position* die Befehls-ID des Menüeintrags.

Flags: Legt fest, wie der Parameter *Position* interpretiert wird. Er kann auf *MF_BYCOMMAND* (Vorgabe) oder auf *MF_BYPOSITION* gesetzt werden.

Rückgabewert Der Rückgabewert ist ungleich Null nach erfolgreicher Ausführung, andernfalls Null.

Hinweis Unabhängig davon, ob das Menü zu einem aktuell angezeigten Fenster gehört, sollte die Anwendung nach einer Veränderung des Menüs die Funktion *DrawMenuBar* aufrufen.

DeleteMetaFile

Syntax
```
BOOL DeleteMetaFile(HANDLE MF);
```

Beschreibung Diese Funktion sperrt den Zugriff auf ein Metafile durch Freigabe der zugehörigen Systemressourcen. Das Metafile selbst wird nicht entfernt, die Funktion macht das Handle des Metafiles (*MF*) ungültig. Der Zugriff auf das Metafile kann somit durch Erstellung eines neuen Handle mit Hilfe der Funktion *GetMetaFile* wieder ermöglicht werden.

MF: Bezeichnet das zu sperrende Handle.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Systemressourcen freigegeben sind und Null, wenn der Parameter *MF* kein gültiges Handle ist.

Querverweis *GetMetaFile*

DeleteObject

Syntax
```
BOOL DeleteObject(HANDLE Handle);
```

Beschreibung Diese Funktion löscht einen virtuellen Stift, Pinsel, Bitmap, eine Region, Schrift oder Farbpalette durch Freigabe des zuge-

hörigen Systemspeichers. Nach Löschung des Objekts ist das Handle *Handle* ungültig.

Handle: Bezeichnet das Handle eines virtuellen Stifts, Pinsels, Bitmaps, einer Region, Schrift oder Farbpalette.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Objekt gelöscht wurde. Er ist Null, wenn der Parameter *Handle* kein gültiges Handle ist oder gerade in einem Gerätekontext benutzt wird.

Hinweis Das zu löschende Objekt sollte aktuell nicht für einen Gerätekontext ausgewählt sein.

Wird ein Schraffurpinsel gelöscht, bleibt das zugehörige Bitmap davon unberührt, es muß selbständig gelöscht werden.

Anwendungen dürfen keinesfalls vordefinierte Objekte löschen.

DestroyCaret

Syntax
```
void DestroyCaret(void);
```

Beschreibung Diese Funktion löscht die Form des Caret, löst es vom aktuell verwaltenden Fenster und löscht es vom Bildschirm, wenn es sichtbar ist. Sie überprüft die Zugehörigkeit des Caret und löscht es nur, wenn es von einem Fenster der aktuellen Task verwaltet wird. War die Form des Caret ein Bitmap, so wird dieses nicht freigegeben. Die Funktion hat keine Parameter.

Hinweis Das Caret stellt eine gemeinsam genutzte Ressource dar. Hat ein Fenster eine bestimmte Form erzeugt, so wird diese gelöscht, wenn das Fenster den Fokus verliert oder deaktiviert wird.

Rückgabewert Keiner.

Querverweis *CreateCaret*

DestroyCursor

Syntax
```
BOOL DestroyCursor(HCURSOR Cursor);
```

Beschreibung Diese Funktion entfernt den zuvor durch *CreateCursor* erzeugten Mauszeiger und gibt den belegten Speicherplatz frei. Sie sollte nicht zur Entfernung von Mauszeigern verwendet werden, die durch andere Funktionen als *CreateCursor* erzeugt wurden.

Cursor: Bezeichnet den zu entfernenden Mauszeiger. Der Mauszeiger darf aktuell nicht verwendet werden.

Rückgabewert Nach erfolgreicher Ausführung ist der Rückgabewert ungleich Null, andernfalls ist er Null.

Querverweis *CreateCursor*

DestroyIcon

Syntax
```
BOOL DestroyIcon(HICON Icon);
```

Beschreibung Diese Funktion entfernt das zuvor durch *CreateIcon* erzeugte Symbol und gibt den belegten Speicherplatz frei. Sie sollte nicht zur Entfernung von Symbolen verwendet werden, die durch andere Funktionen als *CreateIcon* erzeugt wurden.

Icon: Bezeichnet das zu entfernende Symbol. Das Symbol darf aktuell nicht verwendet werden.

Rückgabewert Nach erfolgreicher Ausführung ist der Rückgabewert ungleich Null, andernfalls ist er Null.

Querverweis *CreateIcon*

DestroyMenu

Syntax
```
BOOL DestroyMenu(HMENU Menu);
```

Beschreibung Diese Funktion entfernt das durch *Menu* bezeichnete Menü und gibt den zugehörigen Speicher frei.

Menu: Bezeichnet das zu entfernende Menü.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Menü entfernt wurde, andernfalls Null.

Querverweis *AppendMenu, CreateMenu, InsertMenu*

DestroyWindow

Syntax
```
BOOL DestroyWindow(HWND Wnd);
```

Beschreibung Diese Funktion entfernt das angegebene Fenster. Sie versteckt oder schließt das Fenster dauerhaft, indem sie die entsprechenden Botschaften an das Fenster übergibt und den Fokus entfernt. Außerdem wird das Fenstermenü entfernt und freigegeben, die Anwendungs-Warteschlange geleert, ausstehende Timer gelöscht, der Besitz der Zwischenablage abgegeben und die Kette der Clipboard-Zwischenablage-Viewer unterbrochen, wenn das Fenster an deren Spitze steht. Die Funktion übergibt die Botschaften *WM_DESTROY* und *WM_NCDESTROY* an das Fenster.

Ist das gegebene Fenster anderen Fenstern übergeordnet, so werden seine untergeordneten Fenster ebenfalls entfernt. *DestroyWindow* entfernt zuerst die untergeordneten, dann das angegebene Fenster.

DestroyWindow entfernt auch nicht-modale Dialogfenster, die mit der Funktion *CreateDialog* erzeugt wurden.

Wnd: Bezeichnet das zu entfernende Fenster.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Fenster entfernt wurde, ansonsten ist er Null.

DialogBox

Syntax
```
int DialogBox(HANDLE Instance, LPSTR TemplateName,
HWND WndParent, FARPROC DialogFunc);
```

Beschreibung Diese Funktion erzeugt ein modales Dialogfenster mit dem Typ, der Größe und den Steuerelementen, die in der durch den Parameter *TemplateName* bezeichneten Dialogfenstervorlage beschrieben sind. Der Parameter *WndParent* bezeichnet das Anwendungsfenster, zu dem das Dialogfenster gehört. Botschaften, die das Dialogfenster empfängt, werden von der Callback-Funktion bearbeitet, auf die der Parameter *DialogFunc* zeigt. Die Funktion *DialogBox* gibt die Kontrolle erst dann zurück, wenn die Callback-Funktion das modale Dialogfenster durch den Aufruf der Funktion *EndDialog* schließt.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei die Dialogfenstervorlage enthält.

TemplateName: Zeigt auf einen nullterminierten String, der die Dialogfenstervorlage benennt.

WndParent: Bezeichnet das Fenster, von dem das Dialogfenster verwaltet wird.

DialogFunc: Die Adresse der Prozedurinstanz der Dialogfunktion. Weitere Informationen im folgenden Hinweis.

Rückgabewert Der Rückgabewert bestimmt den Wert des Parameters *Result* der Funktion *EndDialog*. Diese Funktion wird zum Schließen des Dialogfensters verwendet. Die Werte, die vom Dialogfenster zurückgegeben werden, werden von Windows bearbeitet und nicht an die Anwendung übergeben. Der Rückgabewert ist -1, wenn die Funktion das Dialogfenster nicht erstellen konnte.

Hinweis Die Funktion *DialogBox* ruft die Funktion *GetDC* auf, um den Bildschirmkontext zu ermitteln. Es kann zu Problemen kommen, wenn alle Bildschirmkontexte im Bildschirmkontext-Cache von Windows durch *GetDC* zugewiesen wurden und *DialogBox* auf einen weiteren Bildschirmkontext zuzugreifen versucht.

Ein Dialogfenster kann bis zu 255 Steuerelemente enthalten.

Callback-Funktion

```
int FAR PASCAL DialogFunc(HWND Dlg, WORD Msg,
WORD wParam, DWORD lParam);
```

DialogFunc: Der Platzhalter für den in der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Dlg: Bezeichnet das Dialogfenster, das die Botschaft empfängt.

Msg: Bestimmt die Nummer der Botschaft.

wParam: Bestimmt 16 Bits zusätzlicher, botschaftsabhängiger Informationen.

lParam: Bestimmt 32 Bits zusätzlicher, botschaftsabhängiger Informationen.

Rückgabewert Die Callback-Funktion sollte 0 zurückgeben, wenn die Funktion die Botschaft nicht bearbeitet, einen Wert ungleich 0, wenn sie bearbeitet wird.

Hinweis Auch wenn die Callback-Funktion einer Fensterfunktion vergleichbar ist, darf sie nicht die Funktion *DefWindowProc* zur Bearbeitung unerwünschter Botschaften aufrufen. Diese Botschaften werden intern durch die Fensterfunktion der Dialogklasse bearbeitet.

Die Adresse der Callback-Funktion, die im Parameter *DialogFunc* übergeben wird, muß durch die Funktion *MakeProcInstance* erzeugt werden.

DialogBoxIndirect

Syntax

```
int DialogBoxIndirect(HANDLE Instance, HANDLE
DialogTemplate, HWND WndParent, FARPROC DialogFunc);
```

Beschreibung Diese Funktion erzeugt ein modales Dialogfenster mit dem Typ, der Größe und den Steuerelementen, die in der durch den Parameter *TemplateName* bezeichneten Dialogfenstervorlage beschrieben sind. Der Parameter *WndParent* bezeichnet das Anwendungsfenster, welches das Dialogfenster besitzt. Botschaften, die das Dialogfenster empfängt, werden von der Callback-Funktion bearbeitet, auf die der Parameter *DialogFunc* zeigt.

DialogBoxIndirect gibt die Kontrolle erst dann zurück, wenn die Callback-Funktion das modale Dialogfenster durch Aufruf der Funktion *EndDialog* schließt.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei die Dialogfenstervorlage enthält.

DialogTemplate: Bezeichnet einen Block im globalen Speicher, der eine *DLGTEMPLATE*-Datenstruktur enthält.

WndParent: Bezeichnet das Fenster, zu dem das Dialogfenster gehört.

DialogFunc: Die Adresse der Prozedurinstanz der Dialogfunktion. Weitere Informationen im folgenden »Hinweis«.

Rückgabewert Der Rückgabewert bestimmt den Wert des Parameters *Result* der Funktion *EndDialog*. Diese Funktion wird zum Schließen des Dialogfensters verwendet. Die Werte, die vom Dialogfenster zurückgegeben werden, werden von Windows bearbeitet und nicht an die Anwendung übergeben. Der Rückgabewert ist -1, wenn die Funktion das Dialogfenster nicht erstellen konnte.

Hinweis Ein Dialogfenster kann bis zu 255 Steuerelemente enthalten.

Callback-Funktion

```
BOOL FAR PASCAL DialogFunc(HWND Dlg, WORD Msg,
WORD wParam, DWORD lParam);
```

DialogFunc: Der Platzhalter für den in der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Dlg: Bezeichnet das Dialogfenster, welches die Botschaft empfängt.

Msg: Bestimmt die Nummer der Botschaft.

wParam: Bestimmt 16 Bits zusätzlicher, botschaftsabhängiger Informationen.

lParam: Bestimmt 32 Bits zusätzlicher, botschaftsabhängiger Informationen.

Rückgabewert Die Callback-Funktion sollte 0 zurückgeben, wenn die Funktion die Botschaft nicht bearbeitet, einen Wert ungleich 0, wenn sie bearbeitet wird.

Hinweis Auch wenn die Callback-Funktion einer Fensterfunktion vergleichbar ist, darf sie nicht die Funktion *DefWindowProc* zur Bearbeitung unerwünschter Botschaften aufrufen. Diese Botschaften werden intern durch die Fensterfunktion der Dialogklasse bearbeitet.

Die Adresse der Callback-Funktion, die im Parameter *DialogFunc* übergeben wird, muß durch die Funktion *MakeProcInstance* erzeugt werden.

DialogBoxIndirectParam

Syntax

```
int DialogBoxIndirectParam(HANDLE Instance,
HANDLE DialogTemplate, HWND WndParent,
FARPROC DialogFunc, LONG InitParam);
```

Beschreibung Diese Funktion erzeugt ein modales Dialogfenster, übergibt vor dessen Darstellung die Botschaft *WM_INITDIALOG* an die Dialogfunktion und übergibt *InitParam* als Parameter *lParam* dieser Botschaft. Diese Botschaft erlaubt es der Dialogfunktion, die Steuerelemente des Dialogfensters zu initialisieren.

Weitere Informationen über die Erzeugung eines modalen Dialogfensters finden sich in der Beschreibung der Funktion *DialogBoxIndirect*.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei die Dialogfenstervorlage enthält.

DialogTemplate: Bezeichnet einen Block im globalen Speicher, der eine *DLGTEMPLATE*-Datenstruktur enthält.

WndParent: Bezeichnet das Fenster, zu dem das Dialogfenster gehört.

DialogFunc: Ist die Adresse der Prozedurinstanz der Dialogfunktion. Weitere Informationen finden sich in der Beschreibung der Funktion *DialogBoxIndirect* im »Hinweis«.

InitParam: Ein 32-Bit-Wert, der von *DialogBoxIndirectParam* an die Dialogfunktion übergeben wird, wenn diese das Dialogfenster erstellt.

Rückgabewert Der Rückgabewert ist der Wert des Parameters *Result* der Funktion *EndDialog*. Diese Funktion wird zum Schließen des Dialogfensters verwendet. Die Werte, die vom Dialogfenster zurückgegeben werden, werden von Windows bearbeitet und nicht an die Anwendung übergeben. Der Rückgabewert ist -1, wenn die Funktion das Dialogfenster nicht erstellen konnte.

DialogBoxParam

Syntax
```
int DialogBoxParam(HANDLE Instance, LPSTR TemplateName,
HWND WndParent, FARPROC DialogFunc, LONG InitParam);
```

Beschreibung Diese Funktion erzeugt ein modales Dialogfenster, übergibt vor dessen Darstellung die Botschaft *WM_INITDIALOG* an die Dialogfunktion und *InitParam* als Parameter *lParam* dieser Botschaft. Die Botschaft erlaubt es der Dialogfunktion, die Steuerelemente des Dialogfensters zu initialisieren.

Weitere Informationen über die Erzeugung eines modalen Dialogfensters finden sich in der Beschreibung der Funktion *DialogBox*.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei die Dialogfenstervorlage enthält.

TemplateName: Zeigt auf einen nullterminierten String, der die Dialogfenstervorlage benennt.

Parent: Bezeichnet das Fenster, zu dem das Dialogfenster gehört.

DialogFunc: Ist die Adresse der Prozedurinstanz der Dialogfunktion. Weitere Informationen finden sich im Hinweis der Beschreibung der Funktion *DialogBox*.

InitParam: Ein 32-Bit-Wert, der von *DialogBoxParam* an die Dialogfunktion übergeben wird, wenn diese das Dialogfenster erstellt.

Rückgabewert Der Rückgabewert ist der Wert des Parameters *Result* der Funktion *EndDialog*. Diese Funktion wird zum Schließen des Dialogfensters verwendet. Die Werte, die vom Dialogfenster zurückgegeben werden, werden von Windows bearbeitet und nicht an die Anwendung übergeben. Der Rückgabewert ist -1, wenn die Funktion das Dialogfenster nicht erstellen konnte.

DispatchMessage

Syntax
```
LONG DispatchMessage(LPMSG Msg);
```

Beschreibung Diese Funktion übergibt die Botschaft in der durch *Msg* bezeichneten *MSG*-Struktur an die Fensterfunktion des vorgegebenen Fensters.

Msg: Zeigt auf eine *MSG*-Datenstruktur, die Botschaftsinformationen der Windows-Anwendungs-Warteschlange enthält. Die Struktur muß gültige Botschaftswerte enthalten. Steht in *Msg* eine *WM_TIMER*-Botschaft und ist der Parameter *lParam* dieser Botschaft nicht Null, ist dieser Parameter die Adresse einer Funktion, die anstelle der Fensterfunktion aufgerufen wird.

Rückgabewert Der Rückgabewert bestimmt den von der Fensterfunktion zurückgegebenen Wert. Er hängt von der übergebenen Botschaft ab, wird jedoch meist ignoriert.

DlgDirList

Syntax
```
int DlgDirList(HWND Dlg, LPSTR PathSpec, int IDListBox,
int IDStaticPath, WORD FileType);
```

Beschreibung Diese Funktion füllt eine Liste mit Datei- oder Verzeichnisnamen. Die Liste wird durch den Parameter *IDListBox* bezeichnet und mit den Namen der Dateien gefüllt, die dem in *PathSpec* vorgegebenen Pfadnamen entsprechen.

DlgDirList zeigt Unterverzeichnisse in eckige Klammern eingeschlossen ([ ]) und Laufwerke in der Form [-x-], wobei x die Laufwerksbezeichnung darstellt.

Der Parameter *PathSpec* hat die folgende Form:

```
[[Laufwerk:]] [[ [[\u]]Verzeichnis
[[\Verzeichnis]]...\u]] [[Dateiname]]
```

In diesem Beispiel entspricht *Laufwerk:* der Laufwerksbezeichnung, *Verzeichnis* einem gültigen Verzeichnisnamen und *Dateiname* einem gültigen Dateinamen, der mindestens ein Platzhalterzeichen enthalten muß. Platzhalterzeichen sind das Fragezeichen (?, ersetzt ein Zeichen) und der Stern bzw. Asterisk (*, ersetzt beliebig viele Zeichen).

Schließt der Parameter *PathSpec* eine Laufwerksbezeichnung und/oder einen Verzeichnisnamen ein, werden das aktuelle Laufwerk und Verzeichnis entsprechend gewechselt, bevor die Liste gefüllt wird. Das durch *IDStaticPath* bezeichnete Textfeld wird ebenfalls mit der neuen Laufwerksbezeichnung und/oder dem neuen Verzeichnisnamen aktualisiert. Nach dem Füllen einer Liste wird *PathSpec* durch Entfernung der Laufwerksbezeichnung und/oder des Verzeichnisnamens aktualisiert.

DlgDirList sendet die Botschaften *LB_RESETCONTENT* und *LB_DIR* an die Liste.

Dlg: Bezeichnet das Dialogfenster, welches die Liste enthält.

PathSpec: Zeigt auf einen nullterminierten String mit dem Pfadnamen.

IDListBox: Bestimmt den Bezeichner einer Liste. Ist *IDListBox* gleich 0, geht *DlgDirList* davon aus, daß die Liste nicht existiert und versucht nicht, sie zu füllen.

IDStaticPath: Bestimmt den Bezeichner des statischen Textfeldes, der für die Darstellung des aktuellen Laufwerks und Verzeichnisses verwendet wird. Ist *IDStaticPath* 0, geht *DlgDirList* davon aus, daß ein solches Texteld nicht existiert.

FileType: Bestimmt die darzustellenden DOS-Dateiattribute. Möglich ist jede Kombination der im Hinweis beschriebenen Werte. Die Werte können mit Hilfe des bitweisen ODER-Operators (|) kombiniert werden.

Rückgabewert

Der Rückgabewert ist ungleich 0, wenn eine Liste dargestellt wurde, auch wenn es sich dabei um eine leere Liste handelt. Der Rückgabewert 0 zeigt an, daß der Eingabestring keinen gültigen Suchpfad enthielt.

Hinweis

Der Parameter *FileType* bestimmt die DOS-Dateiattribute der darzustellenden Dateien, diese werden in der folgenden Tabelle beschrieben.

Anmerkungen: Ist das Flag *LB_DIR* gesetzt, plaziert Windows die von *DlgDirList* generierten Botschaften in die Anwendungs-Warteschlange. Andernfalls werden sie unmittelbar an die Dialogfunktion übergeben. Ist das Exklusiv-Bit gesetzt,

werden nur Dateien des festgelegten Typs gelistet. Andernfalls werden die festgelegten Dateien zusätzlich zu den anderen Dateien angezeigt.

Attribut	Bedeutung
0x0000	Datendateien für Lese/Schreibzugriff ohne zusätzliche Attribute
0x0001	Schreibgeschützte Dateien
0x0002	Verborgene Dateien
0x0004	Systemdateien
0x0010	Unterverzeichnisse
0x0020	Archiv
0x2000	*LB_DIR*-Flag (vgl. Anmerkung 1)
0x4000	Laufwerke
0x8000	Exklusiv-Bit (vgl. Anmerkung 2)

Querverweis *LB_RESETCONTENT, LB_DIR*

DlgDirListComboBox

Syntax
```
int DlgDirListComboBox(HWND Dlg, LPSTR PathSpec,
int IDComboBox, int IDStaticPath, WORD FileType);
```

Beschreibung Diese Funktion füllt die Liste eines Kombinationsfensters mit Datei- oder Verzeichnisnamen. Die Liste des durch *IDComboBox* bezeichneten Kombinationsfensters wird mit den Namen der Dateien gefüllt, die der Dateinamensvorgabe *PathSpec* entsprechen.

Die Funktion *DlgDirListComboBox* zeigt Unterverzeichnisse in eckige Klammern eingeschlossen ([ ]) und Laufwerke in der Form [-x-], wobei x die Laufwerksbezeichnung darstellt.

Der Parameter *PathSpec* hat die folgende Form:

```
[[Laufwerk:]] [[ [[\u]]Verzeichnis[[\Verzeichnis]]
...\u]] [[Dateiname]]
```

Laufwerk: entspricht der Laufwerksbezeichnung, *Verzeichnis* einem gültigen Verzeichnisnamen und *Dateiname* einem gültigen Dateinamen, der mindestens ein Jokerzeichen enthalten muß. Jokerzeichen sind das Fragezeichen (?, ersetzt ein Zeichen) und der Stern (*, ersetzt beliebig viele Zeichen).

Schließt der Parameter *PathSpec* eine Laufwerksbezeichnung und/oder einen Verzeichnisnamen ein, werden das aktuelle Laufwerk und Verzeichnis entsprechend gewechselt, bevor die Liste gefüllt wird. Das durch *IDStaticPath* bezeichnete Textelement wird ebenfalls mit der neuen Laufwerksbezeichnung und/oder dem neuen Verzeichnisnamen aktualisiert.

Nach dem Füllen der Liste eines Kombinationsfensters wird

PathSpec durch Entfernung der Laufwerksbezeichnung und/oder des Verzeichnisses aktualisiert.

DlgDirListComboBox sendet die Botschaften *CB_RESETCONTENT* und *CB_DIR* an das Kombinationsfenster.

Dlg: Bezeichnet das Dialogfenster, welches das Kombinationsfenster enthält.

PathSpec: Zeigt auf einen nullterminierten String mit dem Verzeichnisnamen.

IDComboBox: Bestimmt den Bezeichner eines Kombinationsfensters. Ist *IDComboBox* gleich 0, geht *DlgDirListComboBox* davon aus, daß das Kombinationsfenster nicht existiert und versucht nicht, die Liste zu füllen.

IDStaticPath: Bestimmt den Bezeichner des statischen Textfeldes, das für die Darstellung des aktuellen Laufwerks und Verzeichnisses verwendet wird. Ist er 0, geht *DlgDirListComboBox* davon aus, daß ein solches Textfeld nicht existiert.

FileType: Bestimmt die darzustellenden DOS-Dateiattribute. Möglich ist jede Kombination der im Hinweis beschriebenen Werte.

Rückgabewert

Der Rückgabewert ist ungleich 0, wenn eine Liste dargestellt wurde, auch wenn es sich dabei um eine leere Liste handelt. Der Rückgabewert 0 zeigt an, daß der Eingabestring keinen gültigen Suchpfad enthielt.

Hinweis

Der Parameter *FileType* bestimmt die DOS-Dateiattribute der darzustellenden Dateien, diese werden in der folgenden Tabelle beschrieben.

Attribut	Bedeutung
0x0000	Datendateien für Lese/Schreibzugriff ohne zusätzliche Attribute
0x0001	Schreibgeschützte Dateien
0x0002	Verborgene Dateien
0x0004	Systemdateien
0x0010	Unterverzeichnisse
0x0020	Archiv
0x2000	*LB_DIR*-Flag (vgl. Anmerkung 1)
0x4000	Laufwerke
0x8000	Exklusiv-Bit (vgl. Anmerkung 2)

Anmerkungen: Ist das Flag *LB_DIR* gesetzt, plaziert Windows die von *DlgDirList* generierten Botschaften in die Anwendungs-Warteschlange. Andernfalls werden sie unmittelbar an die Dialogfunktion übergeben.

Ist das Exklusiv-Bit gesetzt, werden nur Dateien des festge-

legten Typs gelistet. Andernfalls werden die festgelegten Dateien zusätzlich zu den anderen Dateien angezeigt.

Querverweis *CB_RESETCONTENT, CB_DIR*

DlgDirSelect

Syntax
```
BOOL DlgDirSelect(HWND Dlg, LPSTR Str, int IDListBox);
```

Beschreibung Diese Funktion ermittelt die aktuelle Auswahl in einer Liste. Es wird unterstellt, daß die Liste durch die Funktion *DlgDirListComboBox* gefüllt wurde und die Auswahl sich auf eine Laufwerksbezeichnung oder einen Datei- oder Verzeichnisnamen bezieht.

DlgDirSelect kopiert die Auswahl in den durch *Str* bezeichneten Puffer. Wenn sich die aktuelle Auswahl auf eine Laufwerksbezeichnung oder einen Verzeichnisnamen bezieht, entfernt *DlgDirSelect* die einschließenden eckigen Klammern (bei Laufwerksbezeichnungen zusätzlich die Bindestriche), so daß der Name in den neuen Pfadnamen eingefügt werden kann. Liegt keine Auswahl vor, wird *Str* nicht verändert.

DlgDirSelect übergibt die Botschaften *LB_GETCURSEL* und *LB_GETEXT* an die Liste.

Dlg: Bezeichnet das Dialogfenster, das die Liste enthält.

Str: Zeigt auf einen Puffer, der den ausgewählten Pfadnamen aufnehmen soll.

IDListBox: Bestimmt die Integer-ID der Liste in einem Dialogfenster.

Rückgabewert Der Rückgabewert bestimmt den Status der aktuellen Auswahl in der Liste. Handelt es sich dabei um einen Verzeichnisnamen, ist der Rückgabewert ungleich Null, andernfalls Null.

Hinweis Die Funktion *DlgDirSelect* läßt nur die Rückgabe eines einzelnen Dateinamens aus der Liste zu.

Die Liste darf nicht für Mehrfachauswahl vorgesehen sein. Ist letzteres der Fall, wird nicht ungleich Null zurückgegeben und der Parameter *Str* bleibt unverändert.

Querverweis *DlgDirList, LB_GETCURSEL, LB_GETTEXT*

DlgDirSelectComboBox

Syntax
```
BOOL DlgDirSelectComboBox(HWND Dlg, LPSTR Str,
int IDComboBox);
```

Beschreibung Diese Funktion ermittelt die aktuelle Auswahl in der Liste eines Kombinationsfensters. Es wird unterstellt, daß die Liste durch *DlgDirListComboBox* gefüllt wurde und die Auswahl sich auf

eine Laufwerksbezeichnung oder einen Datei- oder Verzeichnisnamen bezieht.

Die Funktion kopiert die Auswahl in den durch *Str* bezeichneten Puffer. Bezieht sich die aktuelle Auswahl auf eine Laufwerksbezeichnung oder einen Verzeichnisnamen, so entfernt sie die einschließenden eckigen Klammern (bei Laufwerksbezeichnungen zusätzlich die Bindestriche), so daß der Name in den neuen Pfadnamen eingefügt werden kann. Liegt keine Auswahl vor, wird *Str* nicht verändert.

DlgDirSelectComboBox übergibt die Botschaften *CB_GETCURSEL* und *CB_GETLBTEXT* an das Kombinationsfenster.

Dlg: Bezeichnet das Dialogfenster, welches das Kombinationsfenster enthält.

Str: Zeigt auf einen Puffer, der den ausgewählten Pfadnamen entnehmen soll.

IDComboBox: Bestimmt die Integer-ID des Kombinationsfensters in einem Dialogfenster.

Rückgabewert Der Rückgabewert bestimmt den Status der aktuellen Auswahl im Kombinationsfenster. Handelt es sich dabei um einen Verzeichnisnamen, ist der Rückgabewert ungleich Null, andernfalls ist er Null.

Hinweis Die Funktion *DlgDirSelectComboBox* läßt nur die Rückgabe eines einzelnen Dateinamens zu.

Querverweis *DlgDirListComboBox, CB_GETCURSEL, DB_GETTEXT*

DPtoLP

Syntax
```
BOOL DPtoLP(HDC DC, LPPOINT Points, int Count);
```

Beschreibung Diese Funktion konvertiert die Punkte eines Geräts in virtuelle Punkte. Sie bildet die Koordinaten jedes durch den Parameter *Points* bezeichneten Punktes aus dem Koordinatensystem des Geräts im virtuellen Koordinatensystem der grafischen Geräteschnittstelle (GDI) ab. Die Konvertierung ist vom aktuellen Abbildungsmodus sowie dem Ursprung und der Ausdehnung von Gerätefenster und Grafikfenster abhängig.

DC: Bezeichnet den Gerätekontext.

Points: Zeigt auf ein Array von Punkten. Jeder Punkt muß eine *POINT*-Datenstruktur sein.

Count: Bestimmt die Anzahl der Punkte im Array.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn alle Punkte konvertiert wurden, andernfalls Null.

DrawFocusRect

Syntax
```
void DrawFocusRect(HDC DC, LPRECT Rect);
```

Beschreibung Diese Funktion zeichnet ein Rechteck im Fokusstil.

DC: Bezeichnet den Gerätekontext.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die Koordinaten des zu zeichnenden Rechtecks enthält.

Hinweis Da es sich hier um eine Exclusives-ODER-Funktion handelt, löscht ein zweiter Aufruf das Rechteck vom Bildschirm.

In dem durch diese Funktion gezeichneten Rechteck kann kein Bildlauf durchgeführt werden. Um einen Bildlauf in einem Bereich durchzuführen, der ein Rechteck enthält, das von dieser Funktion gezeichnet wurde, muß das Rechteck mit Hilfe von *DrawFocusRect* vom Bildschirm gelöscht werden.

Nach erfolgtem Bildlauf muß *DrawFocusRect* erneut aufgerufen werden, um das Rechteck an der neuen Position zu zeichnen.

Rückgabewert Keiner.

DrawIcon

Syntax
```
BOOL DrawIcon(HDC DC, int X, int Y, HICON Icon);
```

Beschreibung Diese Funktion zeichnet ein Symbol im angegebenen Gerät. *DrawIcon* positioniert die obere linke Ecke des Symbols an der durch die Parameter *X* und *Y* bezeichneten Position. Diese Position ist vom aktuellen Abbildungsmodus des Gerätekontextes abhängig.

DC: Bezeichnet den Gerätekontext eines Fensters.

X: Bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Symbols.

Y: Bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Symbols.

Icon: Bezeichnet das zu zeichnende Symbol.

Rückgabewert Nach erfolgreicher Ausführung der Funktion ist der Rückgabewert ungleich Null, andernfalls Null.

Hinweis Die Symbolressource muß vorab mit Hilfe der Funktion *LoadIcon* geladen werden. Vor dem Aufruf dieser Funktion muß der Abbildungsmodus *MM_TEXT* ausgewählt werden.

DrawMenuBar

Syntax
```
void DrawMenuBar(HWND Wnd);
```

Beschreibung Diese Funktion stellt eine Menüleiste wieder her. Sie sollte insbesondere dann aufgerufen werden, wenn eine Menüleiste nach der Erzeugung des besitzenden Fensters verändert wurde.

Wnd: Bezeichnet das Fenster, dessen Menü wiederhergestellt werden muß.

Rückgabewert Keiner.

DrawText

Syntax
```
int DrawText(HDC DC, LPSTR Str, int Count, LPRECT Rect,
WORD Format);
```

Beschreibung Diese Funktion zeichnet formatierten Text in dem durch *Rect* bezeichneten Rechteck. Der Text wird durch Umwandlung von Tabulatoren in entsprechende Leerzeichen, linksbündige, rechtsbündige oder zentrierte Ausrichtung und einen für das Rechteck passenden Zeilenumbruch formatiert. Die Art der Formatierung wird durch den Parameter *Format* festgelegt.

DrawText verwendet zur Textdarstellung die für den Gerätekontext ausgewählte Schrift, Textfarbe und Hintergrundfarbe. Solange das *DT_NOCLIP*-Format nicht verwendet wird, kürzt *DrawText* den Text so, daß er nicht außerhalb des Rechtecks erscheint. Jede Formatierung geht von mehreren Textzeilen aus, außer wenn das Format *DT_SINGLELINE* vorgegeben ist.

DC: Bezeichnet den Gerätekontext.

Str: Zeigt auf den darzustellenden String. Hat der Parameter *Count* den Wert -1, muß der String nullterminiert sein.

Count: Bestimmt die Anzahl der Bytes im String. Ist *Count* -1, wird davon ausgegangen, daß *Str* ein Zeiger auf einen nullterminierten String ist. In diesem Fall berechnet *DrawText* die Anzahl der Zeichen automatisch.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die das Rechteck, in dem der Text formatiert werden soll, in virtuellen Koordinaten enthält.

Format: Bestimmt die Art der Textformatierung. Möglich ist jede Kombination der in Kapitel 4 »Windows-Stile und -Konstanten« in Abschnitt »DT_XXXX« beschriebenen Werte.

Rückgabewert Der Rückgabewert bestimmt die Höhe des Textes.

Hinweis Auch wenn die ausgewählte Schriftart zu groß für das vorge-

gebene Rechteck ist, wird sie von der Funktion *DrawText* nicht durch eine kleinere ersetzt.

Der Abschnitt »DT_XXXX« in Kapitel 4 »Windows-Stile und -Konstanten« beschreibt die Werte des Parameters *Format*. Diese Werte können mit Hilfe des bitweisen ODER-Operators (|) kombiniert werden. Zu beachten ist, daß die Werte *DT_CALERECT*, *DT_EXTERNALLEADING*, *DT_INTERNAL*, *DT_NOCLIP* und *DT_NOPREFIX* nicht zusammen mit dem Wert *DT_TABSTOP* verwendet werden können.

Ellipse

Syntax

```
BOOL Ellipse(HDC DC, int X1, int Y1, int X2, int Y2);
```

Beschreibung Diese Funktion zeichnet eine Ellipse. Der Mittelpunkt der Ellipse wird durch den Mittelpunkt des eingrenzenden Rechtecks – bestimmt durch die Parameter *X1*, *Y1*, *X2* und *Y2* – festgelegt. Der Rahmen der Ellipse wird mit dem aktuellen Stift gezeichnet. Das Innere der Ellipse wird mit dem aktuellen Pinsel ausgefüllt.

Sind die Punkte (*X1*, *Y1*) und (*X2*, *Y2*) identisch, wird nichts gezeichnet.

DC: Bezeichnet den Gerätekontext.

X1, Y1: Bestimmt die virtuelle x- bzw. y-Koordinate der oberen linken Ecke.

X2, Y2: Bestimmt die virtuelle x- bzw. y-Koordinate der unteren rechten Ecke.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Ellipse gezeichnet wurde, andernfalls Null.

Hinweis Die Breite des Rechtecks, gegeben durch die Differenz *X2 - X1*, darf 32 767 Einheiten nicht übersteigen. Dieses Einschränkung gilt auch für die Höhe des Rechtecks (*Y2 - Y1*).

Die aktuelle Position wird von dieser Funktion weder verwendet noch verändert.

EmptyClipboard

Syntax

```
BOOL EmptyClipboard(void);
```

Beschreibung Diese Funktion leert die Zwischenablage und gibt die Handles auf die entsprechenden Daten frei, außerdem wird der Besitz der Zwischenablage dem aufrufenden Fenster zugewiesen.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Zwischenablage leer ist, ist er Null, liegt ein Fehler vor.

Hinweis Die Zwischenablage muß beim Aufruf der Funktion *EmptyClipboard* geöffnet sein.

EnableHardwareInput

Syntax
```
BOOL EnableHardwareInput(BOOL EnableInput);
```

Beschreibung Diese Funktion schaltet Maus- und Tastatureingaben aus. Die Eingaben werden gespeichert, wenn der Parameter *EnableInput* einen Wert ungleich Null hat, ist er Null, werden die Eingaben verworfen.

EnableInput: Veranlaßt die Speicherung der Eingaben, wenn der Parameter *EnableInput* einen Wert ungleich Null ist. Ist er Null, werden die Eingaben verworfen.

Rückgabewert Der Rückgabewert zeigt an, ob die Eingabe über Maus und Tastatur abgeschaltet ist. Er ist ungleich Null (Standardrückgabewert), wenn die Eingabe vorher eingeschaltet war, andernfalls Null.

EnableMenuItem

Syntax
```
BOOL EnableMenuItem(HMENU Menu, WORD IDEnableItem, WORD
Enable);
```

Beschreibung Diese Funktion aktiviert oder deaktiviert Menüoptionen oder versetzt sie in den nicht aktivierbaren Zustand.

Menu: Bestimmt das Menü.

IDEnableItem: Bestimmt die zu markierende Menüoption. Der Parameter *IDEnableItem* kann sich auf Menüpunkte aus Zeilen- und Pop-up-Menüs beziehen.

Enable: Legt die zugehörige Aktion fest. Dies kann eine Kombination von *MF_DISABLED*, *MF_ENABLED* oder *MF_GRAYED* mit *MF_BYCOMMAND* oder *MF_BYPOSITION* sein. Diese Werte können mit Hilfe des bitweisen ODER-Operators (|) kombiniert werden. Die Werte haben folgende Bedeutung:

Wert	Bedeutung
MF_BYCOMMAND	Legt fest, daß der Parameter *IDEnableItem* die Menüpunkt-ID angibt. *MF_BYCOMMAND* ist die Standardeinstellung.
MF_BYPOSITION	Legt fest, daß der Parameter *IDEnableItem* die Position des Menüpunkts angibt (der erste Menüpunkt liegt an der Position 0).
MF_DISABLED	Menüpunkt wird deaktiviert.
MF_ENABLED	Menüpunkt wird aktiviert.

MF_GRAYED	Menüpunkt wird in den nicht aktivierbaren Zustand versetzt.

Rückgabewert Der Rückgabewert definiert den vorhergehenden Menüstatus, er ist -1, wenn der Menüpunkt nicht existiert.

Hinweis In der Beschreibung der Botschaft *WM_SYSCOMMAND* wird erläutert, wie Eingaben in eine Menüleiste verhindert oder ermöglicht werden.

EnableWindow

Syntax
```
BOOL EnableWindow(HWND Wnd, BOOL Enable);
```

Beschreibung Diese Funktion aktiviert bzw. deaktiviert die Eingabe über Maus und Tastatur in das festgelegte Fenster oder Steuerelement. Ist die Eingabe deaktiviert, werden Mausklicks und gedrückte Tasten vom Fenster ignoriert, ist sie aktiviert, werden alle Eingaben bearbeitet.

EnableWindow aktiviert Maus- und Tastatureingaben für das angegebene Fenster, wenn der Parameter *Enable* ungleich Null ist, ist er Null, werden die Eingaben deaktiviert.

Wnd: Bezeichnet das betroffene Fenster.

Enable: Bestimmt, ob das angegebene Fenster aktiviert (ungleich Null) oder deaktiviert (Null) werden soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Fenster korrekt aktiviert bzw. deaktiviert wurde. Ergibt sich ein Fehler, ist er Null.

Hinweis Ein Fenster muß aktiviert werden, bevor es ausgewählt werden kann. Zeigt z.B. eine Anwendung ein nicht-modales Dialogfenster an und ist das Hauptfenster deaktiviert, muß das Hauptfenster vor dem Löschen des Dialogfensters aktiviert werden, da sonst ein anderes Fenster den Fokus erhält. Ist ein untergeordnetes Fenster deaktiviert, wird es von Windows ignoriert, wenn ermittelt wird, welches Fenster Mausbotschaften erhalten soll.

Anfangs sind alle Fenster aktiviert. *EnableWindow* dient zur expliziten Deaktivierung eines Fensters.

EndDeferWindowPos

Syntax
```
void EndDeferWindowPos(HANDLE WinPosInfo);
```

Beschreibung Diese Funktion aktualisiert in einem einzigen Durchlauf gleichzeitig Position und Größe eines oder mehrerer Fenster auf dem Bildschirm. Der Parameter *WinPosInfo* bezeichnet eine

Datenstruktur mit mehreren Fensterpositionen, in der die Aktualisierungsdaten für die Fenster abgelegt sind.

Die Funktion *DeferWindowPos* legt die Aktualisierungsdaten in der Datenstruktur ab, die Funktion *BeginDeferWindowPos* erzeugt zu Beginn die von den anderen Funktionen verwendete Datenstruktur.

WinPosInfo: Bezeichnet eine Datenstruktur für Fensterpositionen mit Informationen über Größe und Position eines oder mehrerer Fenster.

Diese Struktur wird von der Funktion *BeginDeferWindowPos* oder vom letzten Aufruf der Funktion *DeferWindowPos* zurückgegeben.

Rückgabewert Keiner.

Querverweis *BeginDeferWindowPos*

EndDialog

Syntax
```
void EndDialog(HWND Dlg, int Result);
```

Beschreibung Die Funktion schließt ein modales Dialogfenster und übergibt die Eingabewerte an die Funktion *DialogBox*, die das Dialogfenster erzeugt hat. *EndDialog* muß die Bearbeitung immer abschließen, wenn ein modales Dialogfenster mit der Funktion *DialogBox* erzeugt wurde. Sie muß in der Dialogfunktion eines modalen Dialogfensters verwendet werden. Andere Aufrufe der Funktion sind nicht vorgesehen.

Die Dialogfunktion kann *EndDialog* jederzeit aufrufen, auch während der Abarbeitung der Botschaft *WM_INITDIALOG*. Wird sie während der Bearbeitung dieser Botschaft aufgerufen, so wird das Dialogfenster geschlossen, bevor es sichtbar ist bzw. bevor der Fokus gesetzt wird.

EndDialog schließt das Dialogfenster nicht sofort, sondern setzt ein Flag, das das Fenster anweist, unmittelbar nach Beendigung der Dialogfunktion zu schließen. *EndDialog* übergibt die Steuerung an die aufrufende Dialogfunktion, die ihrerseits an Windows übergeben muß.

Dlg: Bezeichnet das zu schließende Dialogfenster.

Result: Enthält den Wert, der vom Dialogfenster an die aufrufende Funktion *DialogBox* zurückgegeben wird.

Rückgabewert Keiner.

EndPaint

Syntax
```
void EndPaint(HWND Wnd, LPPAINTSTRUCT Paint);
```

Beschreibung Die Funktion beendet das Zeichnen im aktuellen Fenster.

EndPaint ist nach jedem Aufruf der Funktion *BeginPaint*, aber erst nach Vollendung der Zeichnung, erforderlich.

Wnd: Legt das neu zu zeichnende Fenster fest.

Paint: Zeigt auf eine *PAINTSTRUCT*-Datenstruktur, in der die durch *BeginPaint* ermittelten Zeichnungsinformationen abgelegt sind.

Hinweis Wurde das Caret von der Funktion *BeginPaint* verborgen, wird es von *EndPaint* wiederhergestellt.

Rückgabewert Keiner.

EnumChildWindows

Syntax
```
BOOL EnumChildWindows(HWND WndParent, FARPROC EnumFunc,
DWORD lParam);
```

Beschreibung Diese Funktion zählt die untergeordneten Fenster des festgelegten übergeordneten Fensters auf, indem das Handle jedes untergeordneten Fensters an die anwendungsunterstützte Callback-Funktion übergeben wird. Die Callback-Funktion wird durch den Zeiger *EnumFunc* bezeichnet.

EnumChildWindows zählt die untergeordneten Fenster auf, bis die aufgerufene Funktion den Wert 0 zurückgibt oder das letzte untergeordnete Fenster aufgelistet wurde.

WndParent: Bestimmt das Fenster, dessen untergeordnete Fenster aufgelistet werden sollen.

EnumFunc: Ist die Adresse der Prozedurinstanz der Callback-Funktion.

lParam: Legt den Wert fest, der an die Callback-Funktion zur Verwendung in der Anwendung übergeben wird.

Rückgabewert Wurden alle untergeordneten Fenster aufgezählt, ist der Rückgabewert ungleich Null, andernfalls ist er Null.

Hinweis Pop-up-Fenster, die zum Parameter *WndParent* gehören, werden nicht aufgezählt.

Die als *EnumFunc* übergebene Adresse muß zuvor durch die Funktion *MakeProcInstance* erzeugt worden sein.

Callback-Funktion
```
BOOL FAR PASCAL EnumFunc(HWND Wnd, DWORD lParam);
```

EnumFunc: Der Platzhalter für den in der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Wnd: Bezeichnet das Handle des Fensters.

lParam: Bestimmt das Argument von *EnumChildWindows*.

Rückgabewert Soll die Aufzählung fortgesetzt werden, muß der Rückgabewert der Callback-Funktion ungleich Null sein. Ein Rückgabewert von 0 stoppt die Auflistung.

EnumClipboardFormats

Syntax
```
WORD EnumClipboardFormats(WORD Format);
```

Beschreibung Die Funktion zählt die zur Zwischenablage gehörigen Formate aus einer Liste der verfügbaren Formate auf. In jedem Funktionsaufruf verweist der Parameter *Format* auf ein bekanntes, verfügbares Format. Zurückgegeben wird das nächste Format aus der Liste. Wird *Format* auf 0 gesetzt, wird das erste Format der Liste abgerufen.

Format: Legt ein bekanntes Format fest.

Rückgabewert Der Rückgabewert kennzeichnet das nächste bekannte Datenformat der Zwischenablage. Er ist 0, wenn *Format* den letzten Eintrag in der Liste bezeichnet oder die Zwischenablage nicht geöffnet ist.

Hinweis Bevor die Formate durch *EnumClipboardFormats* aufgezählt werden können, muß die Zwischenablage mit der Funktion *OpenClipboard* geöffnet werden.

Die Reihenfolge, in der die sendende Anwendung alternative Formate für identische Daten in der Zwischenablage abgelegt hat, entspricht der Reihenfolge, mit der die Auflistung die Formate an die empfangende Anwendung übergibt. Diese Anwendung sollte das erste gelistete Format verwenden, mit dem sie arbeiten kann. Auf diese Weise kann der Sender die Formate empfehlen, die den geringsten Datenverlust zur Folge haben.

Querverweis *OpenClipboard*

EnumFonts

Syntax
```
int EnumFonts(HDC DC, LPSTR FaceName, FARPROC FontFunc,
LPSTR Data);
```

Beschreibung Die Funktion zählt die auf einem bestimmten Gerät zur Verfügung stehenden Schriften auf. Sie findet für jede Schrift, deren Schriftartname durch den Parameter *FaceName* bezeichnet wird, die zugehörigen Informationen und übergibt sie an die Funktion, auf die der Parameter *FontFunc* verweist.

Die in der Anwendung definierte Callback-Funktion kann nun die Schriftinformationen wie gewünscht bearbeiten. Die Aufzählung wird fortgesetzt, bis keine Schriften mehr gefunden werden oder die Callback-Funktion 0 zurückgibt.

DC: Bezeichnet den Gerätekontext.

FaceName: Zeigt auf den nullterminierten String, der den Schriftartnamen der gewünschten Schriften enthält. Ist *FaceName* 0, wählt *EnumFonts* zufällig aus und liefert eine Schrift für jede verfügbare Schriftart.

FontFunc: Die Adresse der Prozedurinstanz der Callback-Funktion. Einzelheiten hierzu finden sich im »Hinweis«.

Data: Zeigt auf die von der Anwendung zur Verfügung gestellten Daten. Die Daten werden zusammen mit den Schriftinformationen an die Callback-Funktion übergeben.

Rückgabewert

Der Rückgabewert ist der letzte Rückgabewert der Callback-Funktion. Seine Bedeutung ist anwenderdefiniert.

Hinweis

Die Adresse, die im Parameter *FontFunc* übergeben wird, muß durch die Funktion *MakeProcInstance* erzeugt werden.

Callback-Funktion

```
int FARPASCAL FontFunc(LPLOGFONT LogFont,
LPTEXTMETRICS TextMetrics, int FontType, LPSTR Data);
```

FontFunc: Platzhalter für den in der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

LogFont: Zeigt auf eine *LOGFONT*-Datenstruktur, die Informationen über die logischen Attribute der Schrift enthält.

TextMetrics: Zeigt auf eine *TEXTMETRIC*-Datenstruktur, die Informationen über die physikalischen Attribute der Schrift enthält.

FontType: Legt die Schriftart fest.

Data: Zeigt auf die anwendungsunterstützen Daten, die von *EnumFonts* übergeben werden.

Rückgabewert

Die Bedeutung des Integer-Rückgabewertes kann gewählt werden.

Hinweis

Der UND-Operator (&) kann zusammen mit den Konstanten *RASTER_FONTTYPE* und *DEVICE_FONTTYPE* zur Bestimmung der Schriftart verwendet werden. Das *RASTER_FONTTYPE*-Bit des Parameters *FontType* legt fest, ob es sich um eine Bitmap- oder eine Vektor-Schrift handelt. Hat das Bit den Wert 1, handelt es sich um eine Bitmap-Schrift, ist der Wert 0, liegt eine Vektor-Schrift vor.

Das *DEVICE_FONTTYPE*-Bit unterscheidet Geräteschriften und Schriften der grafischen Geräteschnittstelle (GDI). Ist der Wert 1, liegt eine Geräteschrift, ist er 0 liegt eine GDI-Schrift vor.

Kann das Gerät Texttransformationen (skalieren, kursiv

usw.) durchführen, wird nur die grundlegende Schrift gelistet. Der Anwender muß prüfen, welche Texttransformationen das jeweilige Gerät durchführen kann, um zu entscheiden, welche zusätzlichen Schriften direkt über das Gerät zu liefern sind. Die GDI kann die Attribute fett, kursiv, unterstrichen und durchgestrichen für jede GDI-Schrift emulieren.

EnumFonts listet nur Schriften aus der internen GDI-Tabelle auf. Schriften, die durch das Gerät selbst erzeugt werden, sind nicht eingeschlossen. Die Möglichkeiten eines Geräts zur Texttransformation können mit der Funktion *GetDeviceCaps* ermittelt werden. Die Information ist über den Index *TextCaps* verfügbar.

Die GDI kann GDI-Bitmap-Schriften horizontal im Verhältnis eins zu fünf und vertikal im Verhältnis eins zu acht skalieren, außer wenn *Proof-Quality* benutzt wird.

EnumMetaFile

Syntax

```
BOOL EnumMetaFile(HDC DC, LOCALHANDLE MF,
FARPROC CallbackFunc, LPBYTE ClientData);
```

Beschreibung

Diese Funktion zählt die GDI-Aufrufe innerhalb des durch den Parameter *MF* bezeichneten Metafiles auf. Sie übergibt jeden GDI-Aufruf innerhalb des Metafiles an die durch den Parameter *CallbackFunc* bezeichnete Funktion. Diese von der Anwendung definierte Callback-Funktion kann jeden GDI-Aufruf in gewünschter Weise abarbeiten.

Die Aufzählung wird fortgesetzt, bis keine GDI-Aufrufe mehr vorhanden sind oder die Callback-Funktion den Wert 0 zurückgibt.

DC: Bezeichnet das dem Metafile zugeordnete Gerät.

MF: Bezeichnet das Metafile.

CallbackFunc: Ist die Adresse der Prozedur der Callback-Funktion. Einzelheiten finden sich im folgenden »Hinweis«.

ClientData: Zeigt auf die Daten, die der Callback-Funktion übergeben werden.

Rückgabewert

Der Rückgabewert ist ungleich Null, wenn die Callback-Funktion alle GDI-Aufrufe in einem Metafile auflistet, andernfalls ist er Null.

Callback-Funktion

```
int FAR PASCAL CallbackFunc(HDC DC,
LPHANDLETABLE HandleTable, LPMETARECORD MFR,
int Obj, BYTE FAR * ClientData);
```

CallbackFunc: Platzhalter für den von der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in

einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

DC: Der das Metafile beinhaltende Gerätekontext.

HandleTable: Zeigt auf eine Handle-Tabelle mit den zugehörigen Objekten (Stifte, Pinsel usw.) im Metafile.

MFR: Zeigt auf einen Metafile-Record im Metafile.

Obj: Bestimmt die Anzahl der Objekte und ihrer Handles in der Handle-Tabelle.

ClientData: Zeigt auf die von der Anwendung bereitgestellten Daten.

Rückgabewert Die Funktion muß ungleich Null zurückgeben, wenn die Auflistung fortgesetzt, Null, wenn sie beendet werden soll.

EnumObjects

Syntax
```
int EnumObjects(HDC DC, int ObjectType,
FARPROC ObjectFunc, LPSTR Data);
```

Beschreibung Diese Funktion listet die verfügbaren Stifte oder Pinsel eines Geräts auf.

Die Callback-Funktion erhält die Informationen für jedes Objekt des vorgegebenen Typs. Die Callback-Funktion wird aufgerufen, bis keine weiteren Objekte mehr auffindbar sind oder der Rückgabewert 0 ist.

DC: Bezeichnet den Gerätekontext.

ObjectType: Bestimmt den Objekttyp. Möglich sind die Werte *OBJ_BRUSH* und *OBJ_PEN*.

ObjectFunc: Die Adresse der Prozedurinstanz der Callback-Funktion. Einzelheiten finden sich im folgenden »Hinweis«.

Data: Zeigt auf die von der Anwendung zur Verfügung gestellten Daten. Die Daten werden zusammen mit den Objektinformationen an die Callback-Funktion übergeben.

Rückgabewert Der Rückgabewert ist der letzte Rückgabewert der Callback-Funktion. Die Bedeutung dieses Wertes wird vom Benutzer definiert.

Hinweis Die Adresse, die durch den Parameter *ObjectFunc* übergeben wird, muß durch die Funktion *MakeProcInstance* erzeugt werden.

Callback-Funktion
```
int FAR PASCAL ObjectFunc(char FAR * LogObject,
char FAR * Data);
```

ObjectFunc: Platzhalter für den von der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in

einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

LogObject: Zeigt auf eine *LogPen*- oder *LogBrush*-Datenstruktur, die Informationen über die virtuellen Attribute des Objekts enthält.

Data: Zeigt auf die Daten der Anwendung, die an die Funktion *EnumObjects* übergeben werden.

Rückgabewert Die Funktion muß ungleich Null zurückgeben, wenn die Auflistung fortgesetzt werden soll.

EnumProps

Syntax
```
int EnumProps(HWND Wnd, FARPROC EnumFunc);
```

Beschreibung Diese Funktion listet alle Einträge der Eigenschaftsliste des betroffenen Fensters auf. Die Auflistung erfolgt durch Übergabe jedes einzelnen Eintrags an die durch *EnumFunc* bezeichnete Callback-Funktion. *EnumProps* arbeitet solange, bis alle Einträge aufgelistet wurden oder die Callback-Funktion 0 zurückgibt.

Wnd: Bezeichnet das Fenster, dessen Eigenschaften aufgelistet werden sollen.

EnumFunc: Ist die Adresse der Prozedurinstanz der Callback-Funktion. Einzelheiten hierzu finden sich im folgenden Hinweis.

Rückgabewert Der Rückgabewert ist der letzte Rückgabewert der Callback-Funktion. Er ist -1, wenn die Funktion keine Liste mit Eigenschaften findet.

Hinweis Eine Anwendung sollte nur Eigenschaften löschen, die von ihr selbst hinzugefügt wurden. Eigenschaftseinträge, die durch andere Anwendungen oder Windows hinzugefügt wurden, sollten keinesfalls gelöscht werden.

Für die Callback-Funktion gelten diese Einschränkungen:

1. Die Callback-Funktion darf die Steuerung weder abgeben noch Abläufe aufrufen, die die Steuerung an andere Tasks abgeben könnten.

2. Der Aufruf der Funktion *RemoveProp* in der Callback-Funktion ist möglich. Die Funktion *RemoveProp* kann jedoch nur Listeneinträge tilgen, die durch die Parameter der Callback-Funktion an diese übergeben wurden.

3. Eine Callback-Funktion sollte nicht versuchen, Listeneinträge hinzuzufügen.

Die Adresse, die durch den Parameter *EnumFunc* übergeben

wird, muß durch die Funktion *MakeProcInstance* erzeugt werden.

Callback-Funktion (Feste Datensegmente)

```
int FAR PASCAL EnumFunc(HWND Wnd, LPSTR Str,
HANDLE Data);
```

In Anwendungen und dynamischen Bibliotheken mit festen Datensegmenten sowie in dynamischen Bibliotheken mit verschiebbaren Datensegmenten, die keinen Stack beinhalten, muß die Callback-Funktion die oben angegebene Form aufweisen.

EnumFunc: Der Platzhalter für den von der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Wnd: Bezeichnet das Handle zu dem Fenster, das die Eigenschaftsliste enthält.

Str: Zeigt auf den nullterminierten String, der dem Daten-Handle zugeordnet wurde, nachdem die Anwendung die Funktion *SetProp* zur Festlegung der Eigenschaften aufgerufen hat. Übergibt die Anwendung statt eines Strings ein Atom an die Funktion *SetProp*, so enthält der Parameter *Str* das Atom im niederwertigen und eine 0 im höherwertigen WORD.

Data: Bezeichnet das Daten-Handle.

Rückgabewert

Die Callback-Funktion kann jede gewünschte Aufgabe ausführen. Soll die Auflistung fortgesetzt werden, muß der Rückgabewert ungleich Null sein. Ist er Null, wird die Auflistung beendet.

Callback-Funktion (verschiebbare Datensegmente)

```
int FAR PASCAL EnumFunc(HWND Wnd, WORD Dummy,
LPSTR Str, HANDLE Data);
```

EnumFunc: Der Platzhalter für den von der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Wnd: Bezeichnet ein Fenster der aktuellen Task mit der Eigenschaftsliste.

Dummy: Blindparameter.

Str: Zeigt auf einen nullterminierten String, der dem Daten-Handle zugeordnet wurde, nachdem die Anwendung die Funktion *SetProp* aufrief, um die Eigenschaft zu setzen. Hat die Anwendung statt eines Strings ein Atom an die Funktion *SetProp* übergeben, so enthält *Str* dieses Atom.

Data: Bezeichnet ein Daten-Handle.

Rückgabewert

Die Callback-Funktion kann jede gewünschte Aufgabe ausfüh-

ren. Soll die Auflistung fortgesetzt werden, muß ein Wert ungleich Null, andernfalls Null zurückgegeben werden.

Hinweis Diese alternative Syntax ist nötig, da die Verschiebung von Datensegmenten die entsprechenden Zeiger ungültig werden läßt. Das Datensegment wird vor allem verschoben, wenn die Callback-Funktion mehr Speicher im lokalen Heap benötigt, als gegenwärtig verfügbar ist.

EnumTaskWindows

Syntax
```
BOOL EnumTaskWindows(HANDLE Task, FARPROC EnumFunc,
LONG lParam);
```

Beschreibung Diese Funktion listet alle Fenster auf, die durch den Rückgabewert der Funktion *GetCurrentTask* bezeichnet werden.

Eine Task ist jedes Programm, das als eigenständige Einheit abläuft. Alle Anwendungen werden als Tasks ausgeführt. Jede Instanz einer Anwendung ist wiederum eine Task.

Die Auflistung wird beendet, wenn die durch den Zeiger *EnumFunc* bezeichnete Callback-Funktion den Wert Null zurückgibt.

Task: Bezeichnet die Task. Die Funktion *GetCurrentTask* gibt dieses Handle zurück.

EnumFunc: Die Adresse der Prozedurinstanz der Callback-Funktion des Fensters.

lParam: 32-Bit-Wert, der zusätzliche Parameter an die durch *EnumFunc* bezeichnete Callback-Funktion übergibt.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn alle zu einer Task gehörenden Fenster aufgelistet wurden, in jedem anderen Fall ist der Rückgabewert Null.

Callback-Funktion
```
BOOL FAR PASCAL EnumFunc(HWND Wnd, DWORD lParam);
```

EnumFunc: Platzhalter für den von der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Wnd: Bezeichnet ein zur aktuellen Task gehörendes Fenster.

lParam: Bestimmt dasselbe Argument, das an die Funktion *EnumTaskWindows* übergeben wurde.

Rückgabewert Die Callback-Funktion kann jede gewünschte Aufgabe ausführen. Soll die Auflistung fortgesetzt werden, muß der Rückgabewert ungleich Null sein. Ist er Null, wird die Auflistung beendet.

Querverweis *GetCurrentTask*

EnumWindows

Syntax
```
BOOL EnumWindows(FARPROC EnumFunc, LONG lParam);
```

Beschreibung Diese Funktion listet alle übergeordneten Fenster auf dem Bildschirm durch Übergabe der zugehörigen Handles dieser Fenster auf. Die Übergabe erfolgt an die durch den Parameter *EnumFunc* bezeichnete Callback-Funktion.

Untergeordnete Fenster werden nicht gelistet. *EnumWindows* fährt mit der Auflistung fort, bis die aufgerufene Funktion den Wert Null zurückgibt oder das letzte Fenster aufgelistet wurde.

EnumFunc: Die Adresse der Prozedurinstanz der *Callback-Funktion.* Einzelheiten hierzu finden sich im folgenden »Hinweis«.

lParam: Legt den Übergabewert an die Callback-Funktion fest.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn alle Fenster gelistet wurden, andernfalls ist er Null.

Hinweis Die Adresse, die durch *EnumFunc* übergeben wird, muß durch die Funktion *MakeProcInstance* erzeugt werden.

Callback-Funktion
```
BOOL FAR PASCAL Enum Func(HWND Wnd, DWORD lParam);
```

EnumFunc: Der Platzhalter für den von der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Wnd: Bezeichnet das Handle des Fensters.

lParam: Bestimmt das 32-Bit-Argument der Funktion *EnumWindows*.

Rückgabewert Der Rückgabewert muß ungleich Null sein, wenn die Auflistung fortgesetzt werden soll. Null stoppt die Auflistung.

EqualRect

Syntax
```
BOOL EqualRect(LPRECT Rect1, LPRECT Rect2);
```

Beschreibung Diese Funktion überprüft die Übereinstimmung zweier Rechtecke, indem sie die Koordinaten der oberen linken und der unteren rechten Ecken vergleicht.

Rect1: Zeigt auf eine *RECT*-Datenstruktur, die die Koordinaten der oberen linken und der unteren rechten Ecke des ersten Rechtecks enthält.

Rect2: Zeigt auf eine *RECT*-Datenstruktur, die die Koordinaten der oberen linken und der unteren rechten Ecke des zweiten Rechtecks enthält.

Rückgabewert Sind die Rechtecke identisch, ist der Rückgabewert ungleich Null, andernfalls ist er Null.

EqualRgn

Syntax
```
BOOL EqualRgn(HRGN SrcRgn1, HRGN SrcRgn2);
```

Beschreibung Diese Funktion überprüft zwei vorgegebene Regionen auf ihre Identität.

SrcRgn1: Bezeichnet die erste Region.

SrcRgn2: Bezeichnet die zweite Region.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Regionen übereinstimmen, andernfalls ist er Null.

Escape

Syntax
```
int Escape(HDC DC, int Escape, int Count, LPSTR InData,
LPSTR OutData);
```

Beschreibung Diese Funktion erlaubt es der Anwendung, auf die Fähigkeiten einer bestimmten Geräteeinheit zuzugreifen, die über die grafische Geräteschnittstelle (GDI) nicht direkt verfügbar sind. *Escape*-Aufrufe einer Anwendung werden übersetzt und an den Gerätetreiber übergeben.

DC: Bestimmt den Gerätekontext.

Escape: Bestimmt die auszuführende *Escape*-Funktion. Die *Escape*-Funktionen werden in Kapitel 4 »Windows-Stile und -Konstanten«, Abschnitt »Drucker Escape-Code« beschrieben.

Count: Bestimmt die Anzahl der Datenbytes, auf die der Parameter *InData* zeigt.

InData: Zeigt auf die von dieser *Escape*-Funktion benötigte Struktur mit den Eingabedaten.

OutData: Zeigt auf die Datenstruktur, welche nach Ausführung der *Escape*-Funktion empfangen wird. Werden keine Daten zurückgegeben, muß der Parameter *OutData* den Wert NULL haben.

Rückgabewert Der Rückgabewert ist positiv, wenn die Funktion erfolgreich ausgeführt wurde. Ausnahme ist die *Escape*-Funktion *QueryEscSupport*, die nur die Implementierung überprüft.

Der Rückgabewert ist 0, wenn die *Escape*-Funktion nicht implementiert wurde, ein negativer Rückgabewert zeigt einen Fehler an.

Die folgende Liste gibt einen Überblick über häufige Fehlermeldungen (vergleiche Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »SP_XXXX«):

Wert	Bedeutung
SP_ERROR	Allgemeiner Fehler.
SP_OUTOFDISK	Speicherplatz auf der Diskette bzw. Platte ist nicht ausreichend für Spooling. Zusätzlicher Speicherplatz kann nicht verfügbar gemacht werden.
SP_OUTOFMEMORY	Arbeitsspeicher ist für Spooling nicht ausreichend.
SP_USERABORT	Der Anwender hat den Druckauftrag mit Hilfe des Druck-Managers abgebrochen.

EscapeCommFunction

Syntax

```
int EscapeCommFunction(int Cid, int Func);
```

Beschreibung Diese Funktion weist die durch den Parameter *Cid* festgelegte serielle Schnittstelle an, die durch den Parameter *Func* bezeichnete erweiterte Funktion auszuführen.

Cid: Legt die serielle Schnittstelle für die Ausführung der erweiterten Funktion fest. Die Funktion *OpenComm* gibt diesen Wert zurück.

Func: Legt den Funktionscode der erweiterten Funktion fest. Folgende Werte stehen zur Verfügung:

Wert	Beschreibung
CLRDTR	Löscht das Data-Terminal-Ready-Signal (DTR).
CLRRTS	Löscht das Request-To-Sent-Signal (RTS).
RESETDEV	Wenn möglich, wird das Gerät in den Ausgangszustand zurückgesetzt.
SETDTR	Sendet das Data-Terminal-Ready-Signal.
SETRTS	Sendet das Request-To-Send-Signal.
SETXOFF	Beeinflußt die Übertragung so, als wäre das Zeichen *XOFF* empfangen worden.
SETXON	Beeinflußt die Übertragung so, als wäre das Zeichen *XON* empfangen worden.

Rückgabewert Der Rückgabewert ist nach erfolgreicher Ausführung der Funktion 0. Wurde durch den Parameter *Func* ein ungültiger Funktionscode festgelegt, ist der Rückgabewert negativ.

Querverweis *OpenComm*

ExcludeClipRect

Syntax

```
int ExcludeClipRect(HDC DC, int X1, int Y1, int X2,
int Y2);
```

Beschreibung Diese Funktion erzeugt eine neue Clipping-Region, die aus der existierenden Clipping-Region abzüglich des festgelegten Rechtecks besteht.

DC: Bezeichnet den Gerätekontext.

X1: Bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Rechtecks.

Y1: Bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Rechtecks.

X2: Bestimmt die virtuelle x-Koordinate der unteren rechten Ecke des Rechtecks.

Y2: Bestimmt die virtuelle y-Koordinate der unteren rechten Ecke des Rechtecks.

Rückgabewert Der Rückgabewert gibt den Typ der neuen Clipping-Region an. Folgende Werte können dabei auftreten:

Wert	Bedeutung
COMPLEXREGION	Die Region hat überlappende Grenzen.
ERROR	Es wurde keine Region erzeugt.
NULLREGION	Die Region ist leer.
SIMPLEREGION	Die Region hat keine überlappenden Grenzen.

Hinweis Die durch den absoluten Wert der Differenz *X2-X1* festgelegte Breite des Rechtecks darf 32 767 Einheiten nicht überschreiten. Diese Grenze gilt auch für die Höhe des Rechtecks (*Y2 - Y1*).

ExcludeUpdateRgn

Syntax

```
int ExcludeUpdateRgn(HDC DC, HWND Wnd);
```

Beschreibung Diese Funktion verhindert das Zeichnen innerhalb ungültiger Bereiche eines Fensters durch den Ausschluß einer aktualisierten Region aus der Clipping-Region des Fensters.

DC: Bezeichnet den zur Clipping-Region gehörenden Gerätekontext.

Wnd: Bezeichnet das zu aktualisierende Fenster.

Rückgabewert Dieser Wert bestimmt den Typ der resultierenden Region. Folgende Werte sind möglich:

Wert	Bedeutung
COMPLEXREGION	Die Region hat überlappende Grenzen.
ERROR	Es wurde keine Region erzeugt.
NULLREGION	Die Region ist leer.
SIMPLEREGION	Die Region hat keine überlappenden Grenzen.

ExitWindows

Syntax
```
BOOL ExitWindows(DWORD Reserved, WORD ReturnCode);
```

Beschreibung Diese Funktion startet die Standardschließprozedur von Windows. Geben die Anwendungen keine Warnungen zurück, wird die Windows-Sitzung beendet und zu DOS zurückgekehrt. Windows übergibt die Botschaft *WM_QUERYENDSESSION* an die Anwendungen, um anzuzeigen, daß Windows beendet werden soll. Stimmen alle Anwendungen zu, sendet Windows die Botschaft *WM_ENDSESSION* an alle Anwendungen, bevor die Arbeitssitzung endgültig beendet wird. Anschließend wird die Funktion 0x4C des DOS-Interrupt 0x21 aufgerufen.

Reserved: Ist reserviert und sollte auf 0 gesetzt werden.

ReturnCode: Legt den Rückgabewert an DOS nach dem Abschluß von Windows fest.

Rückgabewert Der Rückgabewert ist Null, wenn eine oder mehrere Anwendungen das Schließen ablehnen. Stimmen alle Anwendungen zu, kehrt die Funktion nicht zurück.

Querverweis *WM_QUERYENDSESSION, WM_ENDSESSION*

ExtFloodFill

Syntax
```
BOOL ExtFloodFill(HDC DC, int X, int Y, COLORREF Color,
WORD FillType);
```

Beschreibung Diese Funktion füllt einen Bereich der Bildschirmoberfläche mit dem aktuellen Pinsel.

Wenn *FillType* auf *FLOODFILLBORDER* gesetzt wird, geht die Funktion davon aus, daß der gesamte angegebene Bereich von der durch den Parameter *Color* festgelegten Farbe umschlossen ist. Die Funktion beginnt bei dem durch die Parameter *X* und *Y* festgelegten Punkt und füllt das Gebiet in alle Richtungen bis zur angegebenen Farbgrenze.

Wird *FillType* auf *FLOODFILLSURFACE* gesetzt, füllt die Funktion *ExtFloodFill* ausgehend von dem durch *X* und *Y* festgelegten Punkt alle angrenzenden Gebiete, welche die in *Color* festgelegte Farbe enthalten.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate des Ausgangspunktes.

Y: Bestimmt die virtuelle y-Koordinate des Ausgangspunktes.

Color: Bestimmt die Farbe der Begrenzung bzw. die des auszufüllenden Bereichs. Die Interpretation dieses Werts hängt vom Parameter *FillType* ab.

FillType: Bestimmt die Art des Ausmalens. Folgende Werte sind möglich:

Wert	Bedeutung
FLOODFILLBORDER	Der auszumalende Bereich wird von der durch *Color* festgelegten Farbe begrenzt. Diese Art des Ausmalens entspricht der der Funktion *FloodFill*.
FLOODFILLSURFACE	Der auszumalende Bereich wird durch die in *Color* festgelegte Farbe definiert. Das Ausmalen wird nach außen in allen Richtungen solange fortgesetzt, wie die festgelegte Farbe angetroffen wird. Diese Art des Ausmalens ist insbesondere bei mehrfarbigen Rändern hilfreich.

Rückgabewert Der Rückgabewert ist bei erfolgreicher Ausführung der Funktion ungleich 0. In folgenden Fällen ist der Rückgabewert 0:

- Das Ausmalen konnte nicht abgeschlossen werden.
- Der vorgegebene Punkt hat die durch *Color* festgelegte begrenzende Farbe (wenn *FLOODFILLBORDER* angefordert wurde).
- Der vorgegebene Punkt hat nicht die durch *Color* festgelegte Farbe (wenn *FLOODFILLSURFACE* gesetzt wurde).
- Der Punkt liegt außerhalb der Clipping-Region.

Hinweis Die Funktion *ExtFloodFill* wird nur von Gerätekontexten im Arbeitsspeicher und von Geräten mit Rasterdarstellungstechnik unterstützt.

ExtTextOut

Syntax
```
BOOL ExtTextOut(HDC DC, int X, int Y WORD Options,
LPRECT Rect, LPSTR Str, WORD Count, LPINT Dx);
```

Beschreibung Diese Funktion schreibt einen String in der aktuellen Schrift in einen rechteckigen Bereich des Bildschirms. Bei dem Rechteck kann es sich um einen undurchsichtigen Bereich (ausge-

füllt mit der aktuellen Hintergrundfarbe) oder um eine Clipping-Region handeln.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate für den Ursprung des ersten Zeichens des Strings.

Y: Bestimmt die virtuelle y-Koordinate für den Ursprung des ersten Zeichens des Strings.

Options: Bestimmt den Typ des Rechtecks. Der Parameter kann keinen, einen oder beide der folgenden Werte annehmen: *ETO_CLIPPED* und *ETO_OPAQUE*.

Der Wert *ETO_CLIPPED* weist Windows an, den Text an den Rändern des Rechtecks abzuschneiden. *ETO_OPAQUE* führt zur Ausfüllung des Rechtecks mit der aktuellen Hintergrundfarbe.

Rect: Zeigt auf eine Datenstruktur vom Typ *RECT*. Der Parameter *Rect* kann ein NULL-Zeiger sein.

Str: Zeigt auf den angegebenen String.

Count: Bestimmt die Anzahl der Zeichen im String.

Dx: Zeigt auf ein Array von Werten, welche den Abstand zwischen den Ursprüngen benachbarter Buchstaben festlegen. Zum Beispiel trennen *Dx*[*i*] virtuelle Einheiten die Zeichen i und i +1.

Rückgabewert Der Rückgabewert zeigt an, ob der String dargestellt wurde. Ist dies der Fall, so ist der Wert ungleich Null, andernfalls ist er Null.

Hinweis Ist *D*x NULL, wird der Standardzeichenabstand verwendet.

Die Ursprünge der Zeichenzellen und die Inhalte des Arrays, auf das der Parameter *D*x zeigt, werden in virtuellen Einheiten angegeben. Der Ursprung einer Zeichenzelle ist als ihre obere linke Ecke definiert.

Per Voreinstellung wird die aktuelle Position durch diese Funktion weder benutzt noch aktualisiert. Möglich ist jedoch der Aufruf der Funktion *SetTextAlign* durch die Anwendung. Ist dabei der Parameter *Flags* auf *TA_UPDATECP* gesetzt, wird es Windows ermöglicht, die aktuelle Position jederzeit zu verwenden und zu aktualisieren, wenn die Anwendung die Funktion *ExtTextOut* für einen bestimmten Gerätekontext aufruft. Ist dieses Flag gesetzt, werden die *X*- und *Y*-Koordinaten nachfolgender Aufrufe der Funktion *ExtTextOut* von Windows ignoriert.

Querverweis *SetTextAlign*

FatalExit

Syntax

```
void FatalExit(int Code);
```

Beschreibung

Diese Funktion stellt den aktuellen Status von Windows auf dem Debugging-Bildschirm dar (beim Arbeiten mit einem Monitor wird zur Debugger-Oberfläche gewechselt) und fragt Anweisungen zur weiteren Bearbeitung ab. Es wird der Fehlercode *Code* ausgegeben, gefolgt von einer symbolischen Stack-Darstellung, die die Ausführung bis zu diesem Punkt andeutet.

Anwendungen sollten diese Funktion nur zum Testen aufrufen. Der Aufruf in der normalen Version von Windows würde zum Abbruch der Anwendung führen.

Code: Bestimmt den darzustellenden Fehlercode.

Rückgabewert

Keiner.

Hinweis

Die Funktion fordert den Anwender auf, eine Abfrage mit *Abbrechen*, *Unterbrechen* oder *Ignorieren* zu beantworten.

Die Antwort wird wie folgt bearbeitet:

Antwort	Beschreibung
A	»Abbrechen« – beendet Windows.
U	»Unterbrechen« – simuliert einen nicht-maskierbaren Interrupt (NMI) für den Eintritt in den Debugger.
I	»Ignorieren« – kehrt zum Aufrufer zurück.

Die Funktion *FatalExit* dient nur zum Testen.

Anwendungen sollten diese Funktion immer aufrufen, wenn sie einen schweren Fehler feststellen. Alle Ein- und Ausgaben werden über die serielle Standardschnittstelle (AUX) des Computers empfangen bzw. übertragen. Ist ein Debugger installiert, werden sie von diesem verarbeitet.

FillRect

Syntax

```
int FillRect(HDC DC, LPRECT Rect, HBRUSH Brush);
```

Beschreibung

Diese Funktion füllt ein gegebenes Rechteck mit dem angegebenen Pinsel aus, wobei das gesamte Rechteck einschließlich des linken und oberen Randes, jedoch ohne den rechten und unteren Rand, ausgefüllt wird.

DC: Bezeichnet den Gerätekontext.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die virtuellen Koordinaten des auszufüllenden Rechtecks enthält.

Brush: Bezeichnet den vordefinierten Pinsel, der beim Ausfül-

len des Rechtecks benutzt wird. Ist *Brush* ein ungültiges Handle, wird der aktuell gewählte Pinsel verwendet.

Rückgabewert Der Rückgabewert ist vom Typ Integer. Er wird nicht benutzt und hat keine Bedeutung.

Hinweis Der Pinsel muß vorher mit einer der folgenden Funktionen erstellt worden sein: *CreateHatchBrush*, *CreatePatternBrush* oder *CreateSolidBrush*. Er kann auch mit der Funktion *GetStockObject* abgerufen werden.

Beim Ausfüllen des angegebenen Rechtecks schließt die Funktion *FillRect* die rechte und untere Seite des Rechtecks nicht ein. Die GDI füllt ein Rechteck ungeachtet des Abbildungsmodus bis zur rechten Spalte und der untersten Zeile aus, diese sind aber nicht eingeschlossen.

FillRect vergleicht die Werte der Felder *top*, *bottom*, *left* und *right* des bezeichneten Rechtecks. Wenn *bottom* kleiner oder gleich *top* ist bzw. *right* kleiner oder gleich *left* ist, wird das Rechteck nicht gezeichnet.

Querverweis *CreateHatchBrush*, *CreatePatternBrush*, *CreateSolidBrush*, *GetStockObject*

FillRgn

Syntax
```
BOOL FillRgn(HDC DC, HRGN Rgn, HBRUSH Brush);
```

Beschreibung Diese Funktion füllt die durch den Parameter *Rgn* angegebene Region mit dem durch den Parameter *Brush* bezeichneten Pinsel.

DC: Bezeichnet den Gerätekontext.

Rgn: Bezeichnet die Region, die ausgefüllt werden soll. Die Koordinaten für die vorgegebene Region sind in gerätebezogenen Einheiten angegeben.

Brush: Bezeichnet den Pinsel, der benutzt werden soll, um die Region auszufüllen.

Rückgabewert Der Rückgabewert ist *ungleich Null*, wenn die Funktion erfolgreich war, andernfalls ist er Null.

FindAtom

Syntax
```
ATOM FindAtom(LPSTR Str);
```

Beschreibung Diese Funktion durchsucht die Atomtabelle nach dem String, auf den der Parameter *Str* zeigt, und liefert das dem String zugeordnete Atom.

Str: Zeigt auf den nullterminierten String, nach dem gesucht werden soll.

Rückgabewert Der Rückgabewert bezeichnet das Atom, das dem String zugeordnet ist. Er ist NULL, wenn der String nicht in der Tabelle ist.

FindResource

Syntax
```
HANDLE FindResource(HANDLE Instance, LPSTR Name,
LPSTR ResType);
```

Beschreibung Diese Funktion bestimmt die Position einer Ressource in der angegebenen Ressourcendatei. Die Parameter *Name* und *ResType* geben Namen und Typ der Ressource an.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei die Ressource enthält.

Name: Zeigt auf einen nullterminierten String, der den Namen der Ressource enthält.

ResType: Zeigt auf einen nullterminierten String, der die Typbezeichnung der Ressource darstellt. Für vordefinierte Ressourcentypen sollte der Parameter *ResType* einen der folgenden Werte annehmen:

Wert	Bedeutung
RT_ACCELERATOR	Accelerator-Tabelle.
RT_BITMAP	Bitmap-Ressource.
RT_CURSOR	Mauszeigerressource (siehe »Hinweis«).
RT_DIALOG	Dialogfenstervorlage.
RT_FONT	Schriftressource.
RT_FONTDIR	Schriftenverzeichnisressource.
RT_ICON	Symbolressource (siehe »Hinweis«).
RT_MENU	Menüressource.
RT_RCDATA	Benutzerdefinierte Ressource (Rohdaten).
RT_STRING	String-Ressource (siehe »Hinweis«).

Rückgabewert Bei erfolgreicher Ausführung der Funktion bezeichnet der Rückgabewert die benannte Ressource. Er ist NULL, wenn die angeforderte Ressource nicht gefunden werden kann.

Hinweis Eine Anwendung darf die Funktionen *FindResource* und *LoadResource* nicht aufrufen, um Mauszeiger-, Symbol- und Stringressourcen zu laden. Dies geschieht durch Aufruf einer der folgenden Funktionen:

LoadCursor
LoadIcon
LoadString

Eine Anwendung kann die Funktionen *FindResource* und *LoadResource* aufrufen, um andere vordefinierte Ressourcenty-

pen zu laden. Es wird jedoch empfohlen, die entsprechenden Ressourcen durch Aufruf einer der folgenden Funktionen zu laden:

LoadAccelerators
LoadBitmap
LoadMenu

Ist das höherwertige WORD der Parameter *Name* oder *ResType* 0, gibt das niederwertige WORD die Integer-ID des Namens oder des Typs der angegebenen Ressource an. Andernfalls sind die Parameter Zeiger auf nullterminierte Strings.

Wenn das erste Zeichen des Strings ein Nummernzeichen ist (#), stehen die übrigen Zeichen für eine Dezimalzahl, die die Integer-ID des Ressourcennamens oder des Ressourcentyps darstellt. So bedeutet zum Beispiel der String #258 die Integer-ID 258.

Um den Speicherplatzbedarf der von der Anwendung genutzten Ressourcen zu senken, sollten sie sich mittels Integer-ID auf die Ressourcen beziehen, nicht über den Namen.

FindWindow

Syntax
```
HWND FindWindow(LPSTR ClassName, LPSTR WindowName);
```

Beschreibung Diese Funktion gibt das Handle des Fensters zurück, dessen Klasse mit dem Parameter *ClassName* angegeben ist und dessen Überschrift durch *WindowName* bezeichnet wird.

ClassName: Zeigt auf einen nullterminierten String, der die Klassenbezeichnung des Fensters enthält. Ist *ClassName* NULL, gelten alle Klassenbezeichnungen.

WindowName: Zeigt auf einen nullterminierten String, der den Fensternamen enthält (den Text in der Titelleiste des Fensters). Ist *WindowName* NULL, gelten alle Fensternamen.

Rückgabewert Der Rückgabewert bezeichnet das Fenster, das den angegebenen Klassennamen und Fensternamen hat. Er ist NULL, wenn kein solches Fenster gefunden wurde.

FlashWindow

Syntax
```
BOOL FlashWindow(HWND Wnd, BOOL Invert);
```

Beschreibung Diese Funktion läßt ein angegebenes Fenster einmal aufblinken, d.h. das Erscheinungsbild seiner Titelleiste wird so verändert, als ob das Fenster vom inaktiven in den aktiven Zustand übergegangen wäre oder umgekehrt. Normalerweise läßt man ein Fenster aufblinken, um den Benutzer darüber zu informie-

ren, daß das Fenster Aufmerksamkeit verdient, aber den Fokus gegenwärtig nicht hat.

Wnd: Bezeichnet das Fenster, das aufblinken soll. Das Fenster kann geöffnet oder auch zum Symbol verkleinert sein.

Invert: Bestimmt, ob das Fenster aufleuchten oder wieder in seinen ursprünglichen Zustand versetzt werden soll. Ist *Invert ungleich Null*, wird das Fenster von einem Zustand in den anderen versetzt, ist er *Null*, wird das Fenster wieder in seinen ursprünglichen Zustand versetzt (entweder aktiv oder inaktiv).

Rückgabewert Der Rückgabewert gibt den Status des Fensters vor dem Aufruf von *FlashWindow* an. Er ist ungleich Null, wenn das Fenster vor dem Aufruf aktiv war, andernfalls Null.

Hinweis Die Funktion läßt das Fenster nur einmal aufblinken, für fortwährendes Blinken sollte die Funktion einen System-Timer erstellen.

Der Parameter *Invert* sollte nur dann Null sein, wenn das Fenster den Fokus erhält und nicht mehr aufblinken soll; er sollte bei aufeinanderfolgenden Aufrufen ungleich Null sein, solange das Fenster darauf wartet, den Fokus zu erhalten.

Diese Funktion gibt für Symbole immer ungleich Null zurück. Wenn das Fenster als Symbol existiert, läßt *FlashWindow* nur das Symbol aufleuchten; für Fenstersymbole wird *Invert* ignoriert.

FloodFill

Syntax
```
BOOL FloodFill(HDC DC, int X, int Y, COLORREF Color);
```

Beschreibung Diese Funktion füllt einen Bereich auf der Bildschirmfläche mit dem aktuellen Pinsel aus. Sie geht davon aus, daß der Bereich gemäß dem vom Parameter *Color* angegebenen Wert eingegrenzt wird.

Die Funktion beginnt an dem durch die Parameter *X* und *Y* bezeichneten Punkt und fährt dann in alle Richtungen bis zur Farbbegrenzung fort.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate des Punktes, von dem aus das Ausfüllen durchgeführt wird.

Y: Bestimmt die virtuelle y-Koordinate des Punktes, von dem aus das Ausfüllen durchgeführt wird.

Color: Bestimmt die Grenzfarbe.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich ausgeführt wurde. Er ist Null, wenn das Ausfüllen nicht zu Ende geführt werden konnte, der gegebene Punkt selbst die

von *Color* bezeichnete Farbe hat oder der Punkt außerhalb der Clipping-Region liegt.

Hinweis Die Funktion *FloodFill* kann nur bei Speichergerätekontexten und Geräten, die Rasterdarstellungstechnik unterstützen, verwendet werden. Weitere Informationen finden sich in der Beschreibung der Funktion *GetDeviceCaps*.

Querverweis *ExtFloodFill*

FlushComm

Syntax `int FlushComm(int Cid, int Queue);`

Beschreibung Diese Funktion leert die Sende- oder Empfangswarteschlange der durch den Parameter *Cid* angegeben Schnittstelle. Der Parameter *Queue* bestimmt, welche Warteschlange geleert werden soll.

Cid: Bestimmt die Schnittstelle, deren Warteschlange geleert werden soll. Die Funktion *OpenComm* gibt diesen Wert zurück.

Queue: Bestimmt die Warteschlange, die geleert werden soll. Ist *Queue* 0, wird die Sendewarteschlange geleert, ist er 1, wird die Empfangswarteschlange geleert.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls Null.

Querverweis *OpenComm*

FrameRect

Syntax `int FrameRect(HDC DC, LPRECT Rect, HBRUSH Brush);`

Beschreibung Diese Funktion zeichnet einen Rahmen um das durch den Parameter *Rect* bezeichnete Rechteck und verwendet dabei den angegebenen vordefinierten Pinsel. Die Breite und Höhe des Rahmens beträgt immer eine virtuelle Einheit.

DC: Bezeichnet den Gerätekontext des Fensters.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die virtuellen Koordinaten der oberen linken und der unteren rechten Ecke des Rechtecks enthält.

Brush: Bezeichnet den vordefinierten Pinsel, der zum Einrahmen des Rechtecks verwendet werden soll. Ist *Brush* ein ungültiges Handle, wird der aktuell gewählte Pinsel verwendet.

Rückgabewert Wird nicht verwendet.

Hinweis Der Pinsel muß vorher mit einer der Funktionen *CreateHatchBrush*, *CreatePatternBrush* oder *CreateSolidBrush* erstellt

worden sein. Er kann auch mit der Funktion *GetStockObject* abgerufen werden.

Wenn *bottom* kleiner oder gleich *top* ist bzw. *right* kleiner oder gleich *left* ist, wird der Rahmen des Rechtecks nicht gezeichnet.

Querverweis *CreateHatchBrush, CreatePatternBrush, CreateSolidBrush, Rectangle*

FrameRgn

Syntax
```
BOOL FrameRgn(HDC DC, HRGN Rgn, HBRUSH Brush,
int Width, int Height);
```

Beschreibung Diese Funktion zeichnet mit dem durch den Parameter *Brush* bezeichneten Pinsel einen Rahmen um die die durch den Parameter *Rgn* vorgegebene Region.

Der Parameter *Width* gibt die Breite des Rahmens in vertikalen, der Parameter *Height* die Höhe des Rahmens in horizontalen Pinselstrichen an.

DC: Bezeichnet den Gerätekontext.

Rgn: Bezeichnet die Region, die in einen Rahmen eingeschlossen werden soll. Die Koordinaten für die angegebene Region sind in gerätebezogenen Einheiten angegeben.

Brush: Bezeichnet den Pinsel, der benutzt werden soll, um den Rahmen zu zeichnen.

Width: Bestimmt die Breite in vertikalen Pinselstrichen (in virtuellen Einheiten).

Height: Bestimmt die Höhe in horizontalen Pinselstrichen (in virtuellen Einheiten).

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

FreeLibrary

Syntax
```
void FreeLibrary(HANDLE LibModule);
```

Beschreibung Diese Funktion verringert den Referenzzähler des geladenen Bibliothekmoduls um 1.

Wenn der Referenzzähler 0 erreicht, wird der vom Modul belegte Speicherplatz freigegeben.

LibModule: Bezeichnet das geladene Bibliotheksmodul.

Rückgabewert Keiner.

FreeModule

Syntax
```
BOOL FreeModule(HANDLE Module);
```

Beschreibung Diese Funktion verringert den Referenzzähler des geladenen Moduls um 1. Wenn der Referenzzähler 0 erreicht, wird der vom Modul belegte Speicherplatz freigegeben.

Module: Bezeichnet das geladene Modul.

Rückgabewert Wird nicht verwendet.

FreeProcInstance

Syntax
```
void FreeProcInstance(FARPROC Proc);
```

Beschreibung Die durch den Parameter *Proc* bezeichnete Funktion wird mit *FreeProcInstance* aus dem Datensegment, mit dem sie durch die Funktion *MakeProcInstance* verknüpft wurde, entfernt.

Proc: Die Adresse der Prozedurinstanz der freizugebenden Funktion. Sie muß vorher mit der Funktion *MakeProcInstance* erstellt worden sein.

Hinweis Nach der Freigabe einer Prozedurinstanz führen alle Versuche, die Funktion mittels der freigegebenen Adresse der Prozedurinstanz aufzurufen, zu einem nicht behebbaren Fehler.

Querverweis *MakeProcInstance*

FreeResource

Syntax
```
BOOL FreeResource(HANDLE ResData);
```

Beschreibung Diese Funktion löscht eine geladene Ressource aus dem Speicher, indem sie den reservierten Speicherplatz freigibt, welcher von der Ressource belegt war. *FreeResource* gibt die Ressource erst dann tatsächlich frei, wenn der Referenzzähler 0 ist. Dies ist der Fall, wenn *FreeResource* für diese Ressource genauso oft aufgerufen wurde wie *LoadResource*. Dadurch wird sichergestellt, daß die Daten solange im Speicher verbleiben, wie sie von der Anwendung benötigt werden.

ResData: Bezeichnet die Daten, die der Ressource zugeordnet sind. Es wird unterstellt, daß das Handle mit der Funktion *LoadResource* erzeugt wurde.

Rückgabewert Bei erfolgreicher Ausführung der Funktion ist der Rückgabewert Null, er ist ungleich Null, wenn die Ressource nicht freigegeben wurde.

Querverweis *LoadResource*

FreeSelector

Syntax
```
WORD FreeSelector(WORD Selector)
```

Beschreibung Diese Funktion gibt einen Selektor frei, der ursprünglich von den Funktionen *AllocSelector* oder *AllocDStoCSAlias* zugewiesen wurde. Nachdem die Anwendung *FreeSelector* aufgerufen hat, ist der Selektor ungültig und darf nicht weiter verwendet werden.

Selector: Bestimmt den Selektor, der freigegeben werden soll.

Rückgabewert Bei erfolgreicher Ausführung der Funktion ist der Rückgabewert 0, andernfalls ist er der mit *Selector* vorgegebene Selektor.

Hinweis Eine Anwendung sollte diese Funktion nur benutzen, wenn es unbedingt nötig ist. Die Verwendung dieser Funktion widerspricht allen gängigen Windows-Programmiertechniken.

Querverweis *AllocSelector, AllocDStoCSAlias*

GetActiveWindow

Syntax
```
HWND GetActiveWindow(void);
```

Beschreibung Diese Funktion liefert das Handle des aktiven Fensters. Das aktive Fenster ist entweder das mit dem aktuellen Fokus oder das durch die Funktion *SetActiveWindow* explizit aktivierte Fenster. Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bezeichnet das aktive Fenster.

Querverweis *SetActiveWindow*

GetAspectRatioFilter

Syntax
```
DWORD GetAspectRatioFilter(HDC DC);
```

Beschreibung Diese Funktion ermittelt die aktuellen Einstellungen für das Verhältnis von Bildschirmhöhe und -breite. Dieses Verhältnis wird durch die Breite und Höhe der Pixel eines bestimmten Geräts bestimmt. Informationen über dieses Höhen-/Seitenverhältnisses werden für die Herstellung, Auswahl und Darstellung von Schriften benötigt.

Windows stellt einen besonderen Filter für die Auswahl von Schriften zur Verfügung, der die für dieses spezielle Höhen-/Seitenverhältnis geeigneten Schriften aus allen vorhandenen Schriften filtert. Der Filter arbeitet mit dem durch die Funktion *SetMapperFlags* festgelegten Verhältnis.

DC: Bezeichnet den Gerätekontext, der das Höhen-/Seitenverhältnis enthält.

Rückgabewert Der Rückgabewert bestimmt das Höhen-/Seitenverhältnis des aktuellen Filters. Die x-Koordinate des Höhen-/Seitenverhältnisses ist im höherwertigen WORD, die y-Koordinate im niederwertigen WORD abgelegt.

Querverweis *SetMapperFlags*

GetAsyncKeyState

Syntax
```
int GetAsyncKeyState(int Key);
```

Beschreibung Diese Funktion überprüft, ob im Augenblick des Funktionsaufrufs eine Taste gedrückt ist und ob diese Taste seit dem letzten Aufruf von *GetAsyncKeyState* gedrückt wurde.

Key: Legt einen der 256 möglichen virtuellen Tastencodes fest.

Rückgabewert Der Rückgabewert bestimmt, ob die Taste *Key* seit dem letzten Aufruf von *GetAsyncKeyState* gedrückt wurde und ob sie aktuell gedrückt wird.

Ist das höchstwertige Bit gesetzt, wird die Taste gerade gedrückt. Ist das niederwertigste Bit gesetzt, wurde die Taste nach dem vorherigen Aufruf dieser Funktion betätigt.

Querverweis *GetKeyState*

GetAtomHandle

Syntax
```
HANDLE GetAtomHandle(ATOM AnAtom);
```

Beschreibung Die Funktion liefert ein Handle (relativ zum lokalen Heap) für den String, der zu dem durch *AnAtom* bezeichneten Atom gehört.

AnAtom: Bestimmt den Atom-Bezeichner.

Rückgabewert Der Rückgabewert bezeichnet den zum Atom gehörigen String. Er ist Null, wenn kein solches Atom existiert.

GetAtomName

Syntax
```
WORD GetAtomName(ATOM AnAtom, LPSTR Buffer, int Size);
```

Beschreibung Diese Funktion erstellt eine Kopie des zum Parameter *AnAtom* gehörigen Strings in dem Puffer, auf den der Parameter *Buffer* zeigt. *Size* legt die maximale Größe des Puffers fest.

AnAtom: Bezeichnet den aufzufindenden String.

Buffer: Zeigt auf den Puffer, der den String aufnehmen soll.

Size: Bestimmt die maximale Größe des Puffers in Bytes.

Rückgabewert Der Rückgabewert bestimmt die Anzahl von Bytes, die in den

Puffer kopiert wurden. Er ist Null, wenn das angegebene Atom ungültig ist.

GetBitmapBits

Syntax
```
LONG GetBitmapBits(HBITMAP Bitmap, LONG Count,
LPSTR Bits);
```

Beschreibung Diese Funktion kopiert die Bits des angegebenen Bitmaps in den Puffer, auf den der Parameter *Bits* zeigt. Der Parameter *Count* bestimmt die Anzahl der zu kopierenden Bytes. Zur Feststellung des richtigen Werts von *Count* für das angegebene Bitmap sollte die Funktion *GetObject* verwendet werden.

Bitmap: Bezeichnet das Bitmap.

Count: Legt die Anzahl der zu kopierenden Bytes fest.

Bits: Zeiger auf den Puffer, der das Bitmap speichern soll. Das Bitmap ist ein Array von Bytes. Dieses Array entspricht einem horizontalen Raster, das ein Vielfaches von 16 Bit darstellt.

Rückgabewert Der Rückgabewert bestimmt die tatsächliche Anzahl von Bytes in dem Bitmap. Liegt ein Fehler vor, ist er 0.

GetBitmapDimension

Syntax
```
DWORD GetBitmapDimension(HBITMAP Bitmap);
```

Beschreibung Diese Funktion liefert die Breite und Höhe des durch den Parameter *Bitmap* bestimmten Bitmaps. Höhe und Breite des Bitmaps sollten zuvor durch die Funktion *SetBitmapDimension* gesetzt werden.

Bitmap: Bezeichnet das Bitmap.

Rückgabewert Der Rückgabewert gibt Breite und Höhe des Bitmaps in zehntel Millimetern an. Der Wert der Höhe liegt im höherwertigen, der der Breite im niederwertigen WORD. Wurden diese Werte nicht zuvor durch die Funktion *SetBitmapDimension* gesetzt, ist der Rückgabewert NULL.

Querverweis *SetBitmapDimension*

GetBkColor

Syntax
```
DWORD GetBkColor(HDC DC);
```

Beschreibung Die Funktion gibt die aktuelle Hintergrundfarbe des angegebenen Geräts zurück.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert bestimmt den RGB-Farbwert der aktuellen Hintergrundfarbe.

GetBkMode

Syntax
```
int GetBkMode(HDC DC);
```

Beschreibung Diese Funktion liefert den Hintergrundmodus des festgelegten Geräts. Der Hintergrundmodus wird für Texte und Schraffurpinsel sowie für solche Stiftarten, die keine durchgehende Linie ziehen, verwendet.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert gibt den aktuellen Hintergrundmodus an. Möglich sind die Werte *OPAQUE* und *TRANSPARENT* (siehe Kapitel 4 »Windows-Stile und -Konstanten«.)

GetBrushOrg

Syntax
```
DWORD GetBrushOrg(HDC DC);
```

Beschreibung Diese Funktion ermittelt den aktuellen Ursprung des Pinsels im gegebenen Gerätekontext.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert gibt den aktuellen Ursprung des Pinsels an. Die x-Koordinate ist im niederwertigen, die y-Koordinate im höherwertigen WORD abgelegt. Die Koordinaten werden in Geräteeinheiten angegeben.

Hinweis Zu Beginn hat der Ursprung des Pinsels die Koordinaten (0,0).

GetBValue

Syntax
```
BYTE GetBValue(DWORD RGBColor);
```

Beschreibung Dieses Makro ermittelt den blauen Farbanteil eines RGB-Farbwerts.

RGBColor: Bestimmt ein rotes, ein grünes und ein blaues Farbfeld. Diese Felder bestimmen die Intensität der gegebenen Farbe.

Rückgabewert Der Rückgabewert ist ein Byte, das den blauen RGB-Farbwert enthält.

Hinweis Der Wert 0x0FF entspricht der maximalen Intensität eines einzelnen Bytes, 0x000 entspricht der minimalen Intensität eines einzelnen Bytes.

GetCapture

Syntax
```
HWND GetCapture(void);
```

Beschreibung Diese Funktion stellt ein Handle bereit, welches das Fenster bezeichnet, das den Mausfang hat. Es kann jeweils nur ein einzelnes Fenster den Mausfang besitzen, dieses empfängt die Mauseingaben auch dann, wenn sich der Mauszeiger außerhalb seiner Grenzen befindet. Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bezeichnet das Fenster, das den Mausfang hat. Er ist 0, wenn kein Fenster den Mausfang hat.

Hinweis Ein Fenster erhält den Mausfang, wenn sein Handle als Parameter *Wnd* an die Funktion *SetCapture* übergeben wird.

Querverweis *SetCapture*

GetCaretBlinkTime

Syntax
```
WORD GetCaretBlinkTime(void);
```

Beschreibung Diese Funktion liefert die Blinkrate des Caret, die den Zeitraum zwischen dem Aufblinken des Caret in Millisekunden bestimmt. Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert gibt die Blinkrate in Millisekunden an.

GetCaretPos

Syntax
```
void GetCaretPos(LPPOINT Point);
```

Beschreibung Diese Funktion ermittelt die aktuelle Position des Caret in Form von Bildschirmkoordinaten und kopiert sie in die *POINT*-Struktur *Point*.

Point: Zeigt auf eine *POINT*-Struktur, die die Bildschirmkoordinaten des Caret aufnehmen soll.

Hinweis Die Position des Caret wird immer in den Client-Koordinaten des Fensters angegeben, in dem sich das Caret befindet.

Rückgabewert Keiner.

GetCharWidth

Syntax
```
BOOL GetCharWidth(HDC DC, WORD FirstChar,
WORD LastChar, LPINT Buffer);
```

Beschreibung Diese Funktion ermittelt die Breite einzelner Zeichen in einer fortlaufenden Gruppe von Zeichen der aktuellen Schrift. Wird

zum Beispiel mit dem Parameter *FirstChar* das Zeichen *a* und mit dem Parameter *LastChar* das Zeichen *z* angegeben, ermittelt die Funktion *GetCharWidth* die Breiten aller Kleinbuchstaben. Die Werte werden in dem Puffer abgelegt, auf den *Buffer* zeigt.

DC: Bezeichnet den Gerätekontext.

FirstChar: Gibt das erste Zeichen einer fortlaufenden Gruppe von Zeichen der aktuellen Schrift an.

LastChar: Gibt das letzte Zeichen einer fortlaufenden Gruppe von Zeichen der aktuellen Schrift an.

Buffer: Zeigt auf einen Puffer, der die Werte der Breite einer fortlaufenden Gruppe von Zeichen der aktuellen Schrift aufnimmt.

Rückgabewert Bei erfolgreicher Ausführung der Funktion ist der Rückgabewert ungleich Null, andernfalls Null.

Hinweis Existiert ein Zeichen der fortlaufenden Gruppe in einer bestimmten Schrift nicht, wird der Wert der Breite des Standardzeichens zugewiesen.

GetClassInfo

Syntax
```
BOOL GetClassInfo(HANDLE Instance, LPSTR ClassInfo,
LPWNDCLASS WndClass);
```

Beschreibung Diese Funktion liefert Informationen über eine Fensterklasse. Der Parameter *Instance* bezeichnet die Instanz der Anwendung, welche die Fensterklasse erstellt hat, der Parameter *ClassInfo* bezeichnet die Fensterklasse. Findet die Funktion die angegebene Fensterklasse, werden die *WNDCLASS*-Daten, mit der die Fensterklasse registriert wurde, in die durch den Parameter *WndClass* bezeichnete Datenstruktur vom Typ *WNDCLASS* kopiert.

Instance: Bezeichnet die Instanz der die Fensterklasse erzeugenden Anwendung. Zur Ermittlung von Klassen, die durch Windows definiert wurden (beispielsweise Aktionsschalter oder Listen), muß *Instance* auf NULL gesetzt werden.

ClassName: Zeigt auf einen nullterminierten String, der den Namen der gesuchten Klasse enthält.

WndClass: Zeigt auf eine Struktur vom Typ *WNDCLASS*, in welche die Informationen über die Klasse kopiert werden.

Rückgabewert Der Rückgabewert hat den Wert ungleich Null, wenn die Funktion eine passende Klasse gefunden und die Daten erfolgreich kopiert hat. Der Rückgabewert ist Null, wenn keine entsprechende Klasse gefunden wurde.

Hinweis Die Felder *lpszclassName*, *lpszMenuName* und *hInstance* aus der *WNDCLASS*-Datenstruktur werden durch diese Funktion nicht zurückgegeben. Der Menüname ist nicht intern gespeichert und kann daher nicht zurückgegeben werden. Der Klassenname ist bereits bekannt, da er an diese Funktion übergeben wurde. Alle anderen Felder werden mit den Werten aus der Klassenregistrierung zurückgegeben.

GetClassLong

Syntax
```
LONG GetClassLong(HWND Wnd, int Index);
```

Beschreibung Diese Funktion ermittelt den durch *Index* festgelegten Wert vom Typ long aus der *WNDCLASS*-Struktur des durch den Parameter *Wnd* bezeichneten Fensters.

Wnd: Bezeichnet das Fenster.

Index: Bestimmt den Byte-Offset des zu ermittelnden Werts. Auch folgender Wert ist möglich:

Wert	Bedeutung
GCL_WNDPROC	Bestimmt einen Zeiger auf die Fensterfunktion.

Rückgabewert Der Rückgabewert ist der aus der *WNDCLASS*-Struktur ermittelte Wert.

Hinweis Um auf einen bei der Erstellung der Fensterklassenstruktur zusätzlich reservierten Vier-Byte-Wert zuzugreifen, muß ein positiver Byte-Offset für den durch *Index* angegebenen Index verwendet werden. Der erste Vier-Byte-Wert im zusätzlich belegten Speicher liegt bei Offset 0, der nächste bei Offset 4, usw.

GetClassName

Syntax
```
int GetClassName(HWND Wnd, LPSTR ClassName,
int MaxCount);
```

Beschreibung Diese Funktion liefert den Klassennamen des durch den Parameter *Wnd* bezeichneten Fensters.

Wnd: Bezeichnet das Fenster, dessen Klassenname ermittelt werden soll.

ClassName: Zeigt auf einen Puffer, in dem der Klassenname abgelegt wird.

MaxCount: Legt die maximale Anzahl der im Parameter *Class-*

Name abzulegenden Bytes fest. Ist der Name länger, wird er abgeschnitten.

Rückgabewert Der Rückgabewert gibt die tatsächlich in *ClassName* kopierte Zeichenzahl an. Der Wert ist Null, wenn der Klassenname ungültig ist.

GetClassWord

Syntax
```
WORD GetClassWord(HWND Wnd, int Index);
```

Beschreibung Diese Funktion ermittelt das durch *Index* bezeichnete WORD aus der *WNDCLASS*-Struktur. Das zugrundeliegende Fenster wird durch den Parameter *Wnd* festgelegt.

Wnd: Bezeichnet das Fenster.

Index: Bestimmt den Byte-Offset des zu ermittelnden Wertes. Auch folgende Werte sind möglich:

Wert	Bedeutung
GCW_CBCLSEXTRA	Bestimmt, wieviel Bytes zusätzlicher Klasseninformationen existieren. Informationen zum Zugriff auf diesen Speicher finden sich im folgenden »Hinweis«.
GCW_CBWNDEXTRA	Bestimmt, wieviel Bytes zusätzlicher Fensterinformationen existieren. Informationen zum Zugriff auf diesen Speicher finden sich im folgenden »Hinweis«.
GCW_HBRBACKGROUND	Stellt ein Handle für den Hintergrundpinsel zur Verfügung.
GCW_HCURSOR	Stellt ein Handle für den Mauszeiger zur Verfügung.
GCW_HICON	Stellt ein Handle für das Symbol zur Verfügung.
GCW_HMODULE	Stellt ein Handle für das Modul zur Verfügung.
GCW_STYLE	Ermittelt das Stil-Bit der Fensterklasse.

Rückgabewert Der Rückgabewert ist der in der *WNDCLASS*-Struktur gefundene Wert.

Hinweis Um auf Zwei-Byte-Werte zuzugreifen, die bei der Erstellung der Fensterklassenstruktur zusätzlich reserviert wurden, muß ein positiver Byte-Offset für den durch den Parameter *Index* angegebenen Index verwendet werden. Der erste Zwei-Byte-Wert liegt im zusätzlich belegten Speicher bei Offset 0, der folgende bei Offset 2, usw.

GetClientRect

Syntax
```
void GetClientRect(HWND Wnd, LPRECT Rect);
```

Beschreibung Diese Funktion kopiert die Client-Koordinaten des Client-Bereichs eines Fensters in die *RECT*-Struktur *Rect*. Diese Koordinaten bestimmen die obere linke und untere rechte Ecke des Client-Bereichs. Die Koordinaten werden relativ zur oberen linken Ecke dieses Client-Bereichs, die die Koordinaten (0,0) hat, gebildet.

Wnd: Bezeichnet das dem Client-Bereich zugeordnete Fenster.

Rect: Zeigt auf eine *RECT*-Datenstruktur.

Rückgabewert Keiner.

GetClipboardData

Syntax
```
HANDLE GetClipboardData(WORD Format);
```

Beschreibung Diese Funktion liefert Daten aus der Zwischenablage in dem durch den Parameter *Format* angegebenen Format. Die Zwischenablage muß vorher geöffnet werden.

Format: Bestimmt ein Datenformat. Die Beschreibung aller Datenformate ist in Kapitel 4 »Windows-Stile und -Konstanten« in Abschnitt »CF_XXXX« nachzulesen.

Rückgabewert Der Rückgabewert bezeichnet den Speicherblock mit den Daten der Zwischenablage. Der Typ des Handle ist von dem durch *Format* festgelegten Datenformat abhängig. Liegt ein Fehler vor, ist der Rückgabewert NULL.

Hinweis Die verfügbaren Formate können vorab durch die Funktion *EnumClipboardFormats* aufgelistet werden.

Das von *GetClipboardData* zurückgegebene Daten-Handle wird von der Zwischenablage, nicht von der Anwendung kontrolliert. Um eine längere Abhängigkeit von diesem Daten-Handle zu vermeiden, sollte die Anwendung diese Daten sofort kopieren. Die Anwendung sollte das Daten-Handle weder freigeben noch gesperrt lassen.

Windows unterstützt die Textformate *CF_TEXT* und *CF_OEMTEXT*. *CF_TEXT* ist das Standardtextformat der Zwischenablage, *CF_OEMTEXT* wird für das Textformat in Nicht-Windows-Anwendungen verwendet. Wird *GetClipboardData* zum Abrufen von Daten in einem bestimmten Format verwendet und ist nur das andere Format verfügbar, nimmt Windows automatisch vor Übergabe der Daten an die Anwendung die Konvertierung in das gewünschte Format vor.

Enthält die Zwischenablage Daten im Format *CF_PALETTE* (virtuelle Farbpalette), sollte die Anwendung davon ausgehen,

daß alle anderen Daten in der Zwischenablage bezüglich dieser virtuellen Palette realisiert sind.

Querverweis *SetClipboardData*

GetClipboardFormatName

Syntax
```
int GetClipboardFormatName(WORD Format,
LPSTR FormatName, int MaxCount);
```

Beschreibung Diese Funktion ermittelt den Namen des durch den Parameter *Format* bestimmten registrierten Formats aus der Zwischenablage. Der Name wird in den Puffer kopiert, auf den *FormatName* zeigt.

Format: Bestimmt den Typ des zu ermittelnden Formats. Möglich sind die in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »CF_XXXX« beschriebenen Werte.

FormatName: Zeigt auf den Puffer, der den Namen des Formats aufnehmen soll.

MaxCount: Legt die maximale Länge des in den Puffer zu kopierenden Strings in Bytes fest.
Ist der Name länger, werden überzählige Zeichen abgeschnitten.

Rückgabewert Der Rückgabewert gibt die tatsächliche Länge des in den Puffer kopierten Strings an.
Er ist Null, wenn das gewünschte Format nicht existiert oder zu den vordefinierten gehört.

GetClipboardOwner

Syntax
```
HWND GetClipboardOwner(void);
```

Beschreibung Diese Funktion liefert das Handle des Fensters, welches aktueller Besitzer der Zwischenablage ist.
Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bezeichnet das Fenster, das die Zwischenablage besitzt. Ist die Zwischenablage derzeit keinem Fenster zugeordnet, ist der Wert NULL.

Hinweis Die Zwischenablage kann auch dann Daten enthalten, wenn sie momentan keinem Fenster zugeordnet ist.

GetClipboardViewer

Syntax
```
HWND GetClipboardViewer(void);
```

Beschreibung Diese Funktion liefert das Handle des ersten Fensters in der Kette des Zwischenablage-Viewer.
Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bezeichnet das aktuell für die Darstellung der Zwischenablage verantwortliche Fenster. Existiert kein solches Fenster, ist er NULL.

GetClipBox

Syntax
```
int GetClipBox(HDC DC, LPRECT Rect);
```

Beschreibung Diese Funktion liefert die Größe des kleinstmöglichen, die aktuelle Clipping-Grenze umgebenden Rechtecks. Im Parameter *Rect* werden diese Daten gespeichert .

DC: Bezeichnet den Gerätekontext.

Rect: Zeigt auf eine *RECT*-Datenstruktur, in der die Größe des Rechtecks abgelegt wird.

Rückgabewert Der Rückgabewert bestimmt den Typ der Clipping-Region. Folgende Werte sind möglich:

Wert	Bedeutung
COMPLEXREGION	Die Clipping-Region hat überlappende Grenzen.
ERROR	Der Gerätekontext ist ungültig.
NULLREGION	Die Clipping-Region ist leer.
SIMPLEREGION	Die Clipping-Region hat keine überlappende Grenzen.

GetCodeHandle

Syntax
```
HANDLE GetCodeHandle(FARPROC Proc);
```

Beschreibung Diese Funktion ermittelt, welches Code-Segment die Funktion enthält, auf die der Parameter *Proc* zeigt.

Proc: Die Adresse der Prozedurinstanz.

Rückgabewert Der Rückgabewert bezeichnet das Code-Segment, das die bezeichnete Funktion enthält.

Hinweis Ist das die Funktion enthaltende Code-Segment bereits geladen, wird es von der Funktion *GetCodeHandle* als »kürzlich benutzt« markiert, wenn nicht, versucht die Funktion, das Segment zu laden. Eine Anwendung kann daher diese Funktion verwenden, um ein oder mehrere Segment(e) für eine bestimmte Aufgabe im voraus zu laden.

GetCodeInfo

Syntax

```
void GetCodeInfo(FARPROC Proc, LPVOID SegInfo);
```

Beschreibung

Diese Funktion liefert einen Zeiger auf ein Array von 16-Bit-Werten. Das Array enthält Informationen über das Code-Segment, das die Funktion enthält, auf die der Parameter *Proc* zeigt.

Proc: Die Adresse der Funktion im Segment, über das Informationen ermittelt werden sollen. Anstelle der *Segment:Offset*-Adresse kann der Wert auch die Form eines Modul-Handles und einer Segmentnummer haben. Die Funktion *GetModuleHandle* gibt das Handle eines benannten Moduls zurück.

SegInfo: Zeigt auf ein Feld mit vier 32-Bit-Werten, das mit den Informationen über das Code-Segment gefüllt wird.

Die Beschreibung der Werte in diesem Array findet sich im folgenden »Hinweis«.

Rückgabewert

Keiner.

Hinweis

Der Parameter *SegInfo* zeigt auf ein Array mit vier 32-Bit-Werten, das Informationen über Adresse und Größe des Segments und seine Attribute enthält. Die folgende Liste beschreibt die möglichen Werte:

Offset	Beschreibung
0	Bestimmt den Offset des virtuellen Sektors der Daten des Segments relativ zum Beginn der Datei in Bytes. Der Wert 0 bedeutet, daß keine Daten verfügbar sind.
2	Bestimmt die Größe des Segments in der Datei in Bytes. Der Wert 0 entspricht 65536 Bytes (64 KByte).
4	Enthält Flags, welche die Attribute des Segments bestimmen. Die folgende Liste beschreibt diese Flags: *Bit* *Bedeutung* 0-2 Bestimmt den Typ des Segments. Ist das Bit 0 auf 1 gesetzt, liegt ein Datensegment, sonst ein Code-Segment vor. 3 Bestimmt, ob die Daten des Segments wiederholt werden. Dies ist der Fall, wenn dieses Bit auf 1 gesetzt ist. 4 Bestimmt, ob ein verschiebbares oder festes Datensegment vorliegt. Das Segment ist verschiebbar, wenn das Bit auf 1 gesetzt ist. 5 Wird nicht zurückgegeben.

Offset	Beschreibung	
	6	Wird nicht zurückgegeben.
	7	Bestimmt, ob ein Read-Only-Datensegment oder ein Execute-Only-Code-Segment vorliegt. Ist das Bit auf 1 gesetzt und ist das Segment ein Code-Segment, handelt es sich um ein Execute-Only-Code-Segment. Ist das Bit dagegen auf 0 gesetzt und liegt ein Datensegment vor, handelt es sich um ein Read-Only-Datensegment.
	8	Bestimmt, ob Relozierungsinformationen zu dem Segment existieren. Ist das Bit auf 1 gesetzt, sind entsprechende Informationen vorhanden.
	9	Bestimmt, ob das Segment über Debug-Informationen verfügt. Ist das Bit auf 1 gesetzt, existieren solche Informationen.
	10-11	Werden nicht zurückgegeben.
	12-15	Werden nicht zurückgegeben.
6	Bestimmt die absolute Größe des für das Segment reservierten Speichers. Der Wert kann die tatsächliche Größe des Segments übersteigen. Der Wert 0 entspricht 65536 Bytes (64 KByte).	

GetCommError

Syntax

```
int GetCommError(int Cid, COMSTAT FAR * Stat);
```

Beschreibung

Tritt ein Schnittstellenfehler auf, sperrt Windows die Schnittstelle, bis der Fehler durch die Funktion *GetCommError* behoben wird. Die Funktion füllt den Statuspuffer, auf den der Parameter *Stat* zeigt, mit dem aktuellen Status der durch den Parameter *Cid* angegebenen Schnittstelle.

Außerdem wird der Fehlercode, der seit dem letzten Funktionsaufruf von *GetCommError* aufgetreten ist, zurückgeliefert. Hat *Stat* den Wert NULL, wird nur der Fehlercode zurückgegeben.

Cid: Bestimmt, welche Schnittstelle überprüft werden soll. Die Funktion *OpenComm* gibt diesen Wert zurück.

Stat: Zeigt auf die *COMSTAT*-Struktur, die den Gerätestatus aufnehmen soll. Die Struktur enthält Informationen über eine Schnittstelle.

Rückgabewert

Der Rückgabewert bestimmt den Fehlercode der zuletzt ausgeführten Schnittstellenfunktion. Dabei kann es sich um einen einzelnen oder eine Kombination mehrerer Werte aus der folgenden Übersicht handeln.

Wert	Bedeutung
CE_BREAK	Die Hardware stellt eine Abbruchbedingung fest.
CE_CTSTO	Clear-to-send-Unterbrechung (time out). *CTS* ist für die durch *CtsTimeout* festgelegte Dauer gelöscht, während gleichzeitig versucht wird, ein Zeichen zu übertragen.
CE_DNS	Die parallele Schnittstelle ist nicht ausgewählt.
CE_DSRTO	Data-set-ready-Unterbrechung (time out). *DSR* ist für die durch *DsrTimeout* festgelegte Dauer gelöscht, während gleichzeitig versucht wird, ein Zeichen zu übertragen.
CE_FRAME	Die Hardware stellt einen Framing-Fehler fest.
CE_IOE	Ein I/O-Fehler tritt während des Versuchs der Kommunikation mit einer parallelen Schnittstelle auf.
CE_MODE	Der gewünschte Modus wird nicht unterstützt oder der Parameter *Cid* ist ungültig. Ist der Parameter gesetzt, ist dies der einzig mögliche Fehler.
CE_OOP	Der Drucker meldet Papierende.
CE_OVERRUN	Ein Zeichen wird von der Hardware nicht rechtzeitig gelesen. Das Zeichen ist verloren.
CE_PTO	Bei der Kommunikation mit der parallelen Schnittstelle tritt eine Unterbrechung (time out) auf.
CE_RLSDTO	Receive-line-signal-detect-Unterbrechung. *RLSD* ist für die durch *RlsdTimeout* festgelegte Dauer gelöscht, während gleichzeitig versucht wird, ein Zeichen zu übertragen.
CE_RXOVER	Überlauf der Empfangs-Warteschlange. Entweder reicht der Platz in der Eingabe-Warteschlange nicht aus, oder die Übertragung wurde nach Erhalt des *EofChar*-Zeichens fortgesetzt.
CE_RXPARITY	Die Hardware stellt einen Parity-Fehler fest.
CE_TXFULL	Die Übertragungs-Warteschlange ist voll. Gleichzeitig wird versucht, ein Zeichen zur Warteschlange hinzuzufügen.

Querverweis *OpenComm*

GetCommEventMask

Syntax
```
WORD GetCommEventMask(int Cid, int EvtMask);
```

Beschreibung Diese Funktion liefert den Wert der aktuellen Ereignismaske und löscht sie anschließend. *GetCommEventMask* muß verwendet werden, um den Verlust eines Ereignisses zu verhindern.

Cid: Legt die zu untersuchende Schnittstelle fest. Die Funktion *OpenComm* gibt diesen Wert zurück.

EvtMask: Bestimmt, welche Ereignisse unterstützt werden. Eine Liste der Ereigniswerte findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »Schnittstellenereigniswerte«.

Rückgabewert Der Rückgabewert bestimmt den aktuellen Wert der Ereignismaske. Jedes Bit der Ereignismaske zeigt an, ob ein vorgegebenes Ereignis vorgekommen ist. Ist dies der Fall, ist das Bit auf 1 gesetzt.

Querverweis *OpenComm, SetCommEventMask*

GetCommState

Syntax
```
int GetCommState(int Cid, DCB DCB);
```

Beschreibung Diese Funktion füllt den Puffer mit dem Gerätesteuerblock, auf den der Parameter *DCB* zeigt. Die Schnittstelle wird durch den Parameter *Cid* bezeichnet.

Cid: Bestimmt die zu überprüfende Schnittstelle. Die Funktion *OpenComm* gibt diesen Wert zurück.

DCB: Eine *DCB*-Datenstruktur, die den aktuellen Gerätesteuerblock aufnehmen soll. Die Struktur definiert die Steuerungseinstellungen des Geräts.

Rückgabewert Nach erfolgreicher Ausführung der Funktion ist der Rückgabewert 0, im Fall eines Fehlers ist der Rückgabewert negativ.

Querverweis *OpenComm*

GetCurrentPDB

Syntax
```
WORD GetCurrentPDB(void);
```

Beschreibung Diese Funktion gibt die Paragraphenadresse oder den Selektor der aktuellen DOS-Programmdatenbasis (PDB) zurück. Diese wird auch Program Segment Prefix (PSP) genannt.
Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert ist die Paragraphenadresse oder der Selektor der aktuellen PDB.

GetCurrentPosition

Syntax
```
DWORD GetCurrentPosition(HDC DC);
```

Beschreibung Diese Funktion ermittelt die virtuellen Koordinaten der aktuellen Position.

DC: Bezeichnet einen Gerätekontext.

Rückgabewert Der Rückgabewert gibt die aktuelle Position an. Die y-Koordinate liegt im höherwertigen WORD, die x-Koordinate im niederwertigen.

GetCurrentTask

Syntax
```
HANDLE GetCurrentTask(void);
```

Beschreibung Diese Funktion gibt das Handle der gegenwärtig ausgeführten Task zurück.

Die Funktion hat keine Parameter.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt wurde, bezeichnet der Rückgabewert die Task, andernfalls ist er NULL.

GetCurrentTime

Syntax
```
DWORD GetCurrentTime(void);
```

Beschreibung Diese Funktion ermittelt die aktuelle Windows-Zeit. Dies ist die Anzahl von Millisekunden, die seit dem Systemstart vergangen sind.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert ist die aktuelle Zeit in Millisekunden.

Hinweis Die Funktionen *GetCurrentTime* und *GetMessageTime* geben unterschiedliche Zeitangaben zurück. *GetMessageTime* gibt die Windows-Zeit bei Erzeugung der angegebenen Botschaft zurück, nicht die aktuelle Windows-Zeit.

Ensteht in der Systemuhr ein Überlauf, wird sie auf 0 gesetzt.

GetCursorPos

Syntax
```
void GetCursorPos(LPPOINT Point);
```

Beschreibung Diese Funktion ermittelt die aktuelle Mauszeigerposition in Form von Bildschirmkoordinaten. Dann werden diese in die *POINT*-Struktur *Point* kopiert.

Point: Zeigt auf eine *POINT*-Struktur, die die Bildschirmkoordinaten des Mauszeigers aufnehmen soll.

Rückgabewert Keiner.

Hinweis Die Position des Mauszeigers wird immer in Form von Bildschirmkoordinaten angegeben. Sie wird vom Abbildungsmodus des Fensters, das den Mauszeiger enthält, nicht beeinflußt.

GetDC

Syntax
```
HDC GetDC(HWND Wnd);
```

Beschreibung Diese Funktion ermittelt ein Handle für einen Bildschirmkontext für den Client-Bereich eines gegebenen Fensters. Der Bildschirmkontext kann in nachfolgenden Funktionen der grafischen Geräteschnittstelle (GDI) zum Zeichnen im Client-Bereich verwendet werden.

GetDC stellt einen gemeinsamen, einen klassenspezifischen oder einen privaten Bildschirmkontext in Abhängigkeit vom Klassenstil des vorgegebenen Fensters zur Verfügung. Für gemeinsame Bildschirmkontexte weist *GetDC* dem Kontext bei jedem erneuten Aufruf Standardattribute zu. Bei klassenspezifischen und privaten Kontexten werden die vorher zugewiesenen Attribute nicht verändert.

Wnd: Bezeichnet das Fenster, dessen Bildschirmkontext geliefert werden soll.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt wurde, bezeichnet der Rückgabewert den Bildschirmkontext des Client-Bereichs des gegebenen Fensters, andernfalls ist er NULL.

Hinweis Nach dem Zeichnen mit einem gemeinsamen Bildschirmkontext muß die Funktion *ReleaseDC* verwendet werden, um den Kontext freizugeben.

Klassenspezifische und private Bildschirmkontexte müssen nicht freigegeben werden. Da zu einem bestimmten Zeitpunkt höchstens fünf Bildschirmkontexte verfügbar sind, ist die Freigabe der Kontexte nötig, um anderen Anwendungen den Zugriff auf einen Bildschirmkontext zu ermöglichen.

Querverweis *GetWindowDC*, *ReleaseDC*

GetDCOrg

Syntax
```
DWORD GetDCOrg(HDC DC);
```

Beschreibung Diese Funktion ermittelt den Ursprung des Bildschirmkontextes. Dieser Ursprung bestimmt den von Windows verwendeten Offset zur Übersetzung von Gerätekoordinaten in Client-Koordinaten von Punkten in einem Anwendungsfenster. Der Ursprung ist relativ zum physikalischen Ursprung des Bildschirms gebildet.

DC: Bezeichnet den Gerätekontext, dessen Ursprung ermittelt werden soll.

Rückgabewert Der Rückgabewert gibt den Ursprung in Gerätekoordinaten an. Die y-Koordinate liegt im höherwertigen, die x-Koordinate im niederwertigen WORD.

GetDesktopWindow

Syntax
```
HWND GetDesktopWindow(void);
```

Beschreibung Diese Funktion liefert das Fenster-Handle für das Desktop-Fenster von Windows. Das Desktop-Fenster bedeckt den gesamten Bildschirm und ist die Oberfläche, auf der alle anderen Fenster und Symbole dargestellt werden.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bezeichnet das Desktop-Fenster von Windows.

GetDeviceCaps

Syntax
```
int GetDeviceCaps(HDC DC, int Index);
```

Beschreibung Diese Funktion liefert gerätespezifische Informationen über einen gegebenen Bildschirm.

Der Parameter *Index* legt den Typ der gewünschten Informationen fest.

DC: Bezeichnet den Gerätekontext.

Index: Bestimmt die zurückzugebende Information. Es kann sich um einen beliebigen Wert der im »Hinweis« aufgeführten Tabelle handeln.

Rückgabewert Der Rückgabewert gibt den Wert der gewünschten Information an.

Hinweis Die folgende Tabelle beschreibt die möglichen Werte des Parameters *Index*.

Index	Bedeutung	
DRIVERVERSION	Versionsnummer, z.B. 0x100 für 1.0.	
TECHNOLOGY	Gerätetechnik. Folgende Werte sind möglich:	
	DT_PLOTTER	Vektorplotter
	DT_RASDISPLAY	Rasterbildschirm
	DT_RASPRINTER	Rasterdrucker
	DT_RASCAMERA	Rasterkamera
	DT_CHARSTREAM	Zeichen-Stream
	DT_METAFILE	Metafile
	DT_DISPFILE	Bildschirmdatei
HORZSIZE	Breite der physikalischen Anzeige in Millimeter.	
VERTSIZE	Höhe der physikalischen Anzeige in Millimeter.	
HORZRES	Breite der Anzeige in Pixel.	
VERTRES	Höhe der Anzeige in Scan-Zeilen.	
LOGPIXELSX	Anzahl der Pixel pro virtuellem Zoll über die Breite der Anzeige.	

Index	Bedeutung
LOGPIXELSY	Anzahl der Pixel pro virtuellem Zoll über die Höhe der Anzeige.
BITSPIXEL	Anzahl benachbarter Farb-Bits eines Pixels.
PLANES	Anzahl der Farbebenen.
NUMBRUSHES	Anzahl der gerätespezifischen Pinsel.
NUMMARKERS	Anzahl der Markierungssymbole.
NUMPENS	Anzahl der gerätespezifischen Stifte.
NUMFONTS	Anzahl der gerätespezifischen Schriften.
NUMCOLORS	Anzahl der Einträge in der Farbtabelle des Geräts.
ASPECTX	Relative Breite des zum Zeichnen einer Linie verwendeten Pixels eines Geräts.
ASPECTY	Relative Höhe des zum Zeichnen einer Linie verwendeten Pixels eines Geräts.
ASPECTXY	Diagonale Abmessung des zum Zeichnen einer Linie verwendeten Pixels eines Geräts.
PDEVICESIZE	Benötigte Größe des Geräte-Deskriptors.
CLIPCAPS	Flag, das die Clipping-Fähigkeit des Geräts anzeigt. Er ist 1, wenn das Gerät entlang eines Rechtecks abschneiden kann und 0, wenn es dazu nicht in der Lage ist.
SIZEPALETTE	Anzahl der Einträge in der Systempalette. Dieser Index ist nur dann gültig, wenn der Gerätetreiber das *RC_PALETTE*-Bit im *RASTERCAPS*-Index setzt. Er ist nur verfügbar, wenn die Versionsnummer des Treibers 3.0 oder höher ist.
NUMRESERVED	Anzahl der reservierten Einträge in der Systempalette. Dieser Index ist nur dann gültig, wenn der Gerätetreiber das *RC_PALETTE*-Bit im *RASTERCAPS*-Index setzt. Er ist nur verfügbar, wenn die Versionsnummer des Treibers 3.0 oder höher ist
COLORRES	Gegenwärtige Farbauflösung des Geräts in Bits je Pixel. Dieser Index ist nur gültig, wenn der Gerätetreiber das *RC_PALETTE*-Bit im *RASTERCAPS*-Index setzt. Er ist nur verfügbar, wenn die Versionsnummer des Treibers 3.0 oder höher ist.
RASTERCAPS	Wert, der die Rasterfähigkeiten des Geräts anzeigt. Folgende Werte sind möglich: *RC_BANDING* — Benötigt Frequenzunterstützung. *RC_BITBLT* — Fähig zur Bitmap-Übertragung. *RC_BIGFONT* — Unterstützt Schriften mit mehr als 64 KByte.

Index	Bedeutung	
	RC_BITMAP64	Fähig zur Unterstützung von Bitmaps mit mehr als 64 KByte.
	RC_DI_BITMAP	Fähig zur Unterstützung von *SetDIBits* und *GetDIBits*.
	RC_DIBTODEV	Fähig zur Unterstützung von *SetDIBitsToDevice*.
	RC_FLOODFILL	Fähig zum Ausmalen.
	RC_GDI20_OUTPUT	Fähig zur Unterstützung von Merkmalen der Version Windows 2.0.
	RC_PALETTE	Gerät unterstützt Farbpaletten.
	RC_SCALING	Fähigkeit zum Skalieren.
	RC_STRETCHBLT	Fähig zur Durchführung von *StretchBlt*.
	RC_STRETCHDIB	Fähig zur Durchführung von *StretchDIBits*.
CURVECAPS	Zeigt die Bogenzeichnungsfähigkeit des Geräts an. Die Konstanten haben folgende Bedeutung:	
	CC_CHORD	Gerät kann Kreisbögen darstellen.
	CC_CIRCLES	Gerät kann Kreise darstellen.
	CC_ELLIPSES	Gerät kann Ellipsen darstellen.
	CC_INTERIORS	Gerät kann Ausfüllungen darstellen.
	CC_NONE	Gerät hat keine Bogenzeichnungsfähigkeiten.
	CC_PIE	Gerät kann Tortendiagramme darstellen.
	CC_STYLED	Gerät kann verschiedene Linien darstellen.
	CC_WIDE	Gerät kann breite Linien darstellen.
	CC_WIDESTYLED	Gerät kann verschiedene breite Linien darstellen.
LINECAPS	Zeigt die Linienzeichnungsfähigkeiten des Geräts. Die Konstanten haben folgende Bedeutung:	
	LC_INTERIORS	Gerät kann Ausfüllungen darstellen
	LC_MARKER	Gerät kann Markierungen darstellen.
	LC_NONE	Gerät hat keine Linienzeichnungsfähigkeiten.

Index	Bedeutung	
	LC_POLYLINE	Gerät kann Linienzüge darstellen.
	LC_POLYMARKER	Gerät kann Punktmarkierungen darstellen.
	LC_STYLED	Gerät kann verschiedene Linien darstellen.
	LC_WIDE	Gerät kann breite Linien darstellen.
	LC_WIDESTYLED	Gerät kann verschiedene breite Linien darstellen.
POLYGONALCAPS	Zeigt die Fähigkeit des Geräts zum Zeichnen von Vielecken an. Die Konstanten haben folgende Bedeutung:	
	PC_INTERIORS	Gerät kann Ausfüllungen darstellen.
	PC_NONE	Vielecke können nicht dargestellt werden.
	PC_POLYGON	Gerät kann Vielecke darstellen.
	PC_RECTANGLE	Gerät kann Rechtecke darstellen.
	PC_SCANLINE	Gerät kann Scan-Zeilen darstellen.
	PC_STYLED	Gerät kann verschiedene Rahmen darstellen.
	PC_TRAPEZOID	Gerät kann Trapeze darstellen.
	PC_WIDE	Gerät kann breite Rahmen darstellen.
	PC_WIDE STYLED	Gerät kann verschiedene breite Rahmen darstellen.
	PC_WINDPOLYGON	Gerät kann komplexe Vielecke darstellen.
TEXTCAPS	Zeigt die Textdarstellungsfähigkeiten des Geräts an. Die Konstanten haben folgende Bedeutung:	
	TC_CP_STROKE	Gerät kann vektoriell mit Clipping ausgeben.
	TC_CR_90	Gerät kann Buchstaben um 90 Grad rotieren.
	TC_CR_ANY	Gerät kann beliebige Buchstabenrotationen vornehmen.
	TC_CA_DOUBLE	Gerät kann Zeichen fett setzen.
	TC_LA_ABLE	Gerät kann kursive Zeichen darstellen.

Index	Bedeutung	
	TC_OP_STROKE	Gerät kann vektoriell ausgeben.
	TC_RA_ABLE	Gerät kann Rasterschriften darstellen.
	TC_SA_DOUBLE	Gerät kann mit zweifachem Zeichenabstand skalieren.
	TC_SA_INTEGER	Gerät kann das Mehrfache eines Integerwerts zur Skalierung verwenden.
	TC_SA_CONTIN	Gerät kann einen beliebigen vielfachen Wert zur Skalierung verwenden.
	TC_ST_X_YINDEP	Gerät kann unabhängig von X und Y skalieren.
	TC_SO_ABLE	Gerät kann Zeichen durchgestrichen darstellen.
	TC_UA_ABLE	Gerät kann unterstreichen.
	TC_VA_ABLE	Gerät kann Vektorschriften darstellen.

GetDialogBaseUnits

Syntax

```
LONG GetDialogBaseUnits(void);
```

Beschreibung

Diese Funktion liefert die von Windows zur Erstellung von Dialogfenstern verwendeten Dialogbasiseinheiten, die von der Anwendung zur Berechnung der durchschnittlichen Zeichenbreite einer Schrift verwendet werden sollten.

Die Funktion hat keine Parameter.

Rückgabewert

Der Rückgabewert gibt die Dialogbasiseinheiten an. Das höherwertige WORD enthält die Höhe in Pixel, abgeleitet aus der Höhe der Systemschrift. Das niederwertige WORD enthält die Breite in Pixel, abgeleitet aus der Breite der Systemschrift.

Hinweis

Die zurückgegebenen Werte repräsentieren Dialogbasiseinheiten vor der Skalierung in aktuelle Dialogeinheiten.

Um die tatsächliche Höhe (H) und Breite (B) eines Steuerelements in Pixeln festzustellen, muß folgende Formel mit vorgegebener Höhe (*x*) und Breite (*y*) in Dialogeinheiten sowie der Rückgabewert (Dialogbasiseinheit) der Funktion *GetDialogBaseUnits* verwendet werden:

B = (*x* * *LOWORD*(*Dialogbasiseinheit*))/4
H = (*y* * *HIWORD*(*Dialogbasiseinheit*))/8

Um Rundungsfehler zu vermeiden, sollte die Multiplikation vor der Division durchgeführt werden, außer wenn die Dialogbasiseinheiten ohne Rest durch vier teilbar sind.

GetDIBits

Syntax

```
int GetDIBits(HDC DC, HANDLE Bitmap, WORD StartScan,
WORD NumScans, LPSTR Bits, LPBITMAPINFO BitInfo,
WORD Usage);
```

Beschreibung

Diese Funktion ermittelt die Bits des angegebenen Bitmaps und kopiert sie im geräteunabhängigen Format in den Puffer, auf den der Parameter *Bits* zeigt. *BitInfo* bezeichnet die Farbinformationen der geräteunabhängigen Bits in der *BITMAPINFO*-Datenstruktur.

DC: Legt den Gerätekontext fest.

Bitmap: Legt das Bitmap fest.

StartScan: Bestimmt die Anordnung der ersten Scan-Zeile in dem durch *Bits* bezeichneten Bitmap.

NumScans: Bestimmt die Anzahl der zu kopierenden Zeilen.

Bits: Zeigt auf den Puffer, der die Bitmap-Daten im geräteunabhängigen Format aufnimmt.

BitInfo: Zeigt auf eine *BITMAPINFO*-Datenstruktur, die das Farbformat und die Dimensionierung des geräteunabhängigen Bitmaps enthält.

Usage: Bestimmt, ob die Felder *bmiColors* des Parameters *BitInfo* Indexeinträge der aktuellen virtuellen Farbpalette oder explizite RGB-Werte enthalten.

Der Parameter *Usage* muß einen der folgenden Werte annehmen:

Wert	Bedeutung
DIB_PAL_COLORS	Die Farbtabelle muß aus einem Array von 16-Bit-Indizes bestehen, die in die aktuell realisierte virtuelle Farbpalette verweisen.
DIB_RGB_COLORS	Die Farbtabelle enthält explizite RGB-Werte.

Rückgabewert

Der Rückgabewert bestimmt die Anzahl der Scan-Zeilen, die aus dem Bitmap kopiert wurden. Er ist Null, wenn ein Fehler aufgetreten ist.

Hinweis

Hat der Parameter *Bits* den Wert NULL, füllt *GetDIBits* die *BITMAPINFO*-Datenstruktur aus, auf die *BitInfo* zeigt, die Bits aus dem Bitmap werden aber nicht gespeichert.

Das durch den Parameter *Bitmap* bezeichnete Bitmap darf

durch die Anwendung während dieses Funktionsaufrufs in keinen Gerätekontext ausgewählt sein.

Ursprung des geräteunabhängigen Bitmaps ist dessen untere linke Ecke. Die obere linke Ecke ist der Ursprung, wenn der Abbildungsmodus auf *MM_TEXT* gesetzt ist.

GetDlgCtrlID

Syntax
```
int GetDlgCtrlID(HWND Wnd);
```

Beschreibung Diese Funktion gibt den ID-Wert des durch den Parameter *Wnd* bezeichneten, untergeordneten Fensters zurück.

Wnd: Bezeichnet das untergeordnete Fenster.

Rückgabewert Ist die Funktion erfolgreich, entspricht der Rückgabewert dem numerischen Bezeichner des untergeordneten Fensters. Wenn sie fehlschlägt oder der Parameter *Wnd* kein gültiges Fenster-Handle darstellt, ist der Rückgabewert 0.

Hinweis Da die Fenster der Hauptebene keinen ID-Wert haben, ist der Rückgabewert ungültig, wenn der Parameter *Wnd* auf ein solches Fenster verweist.

GetDlgItem

Syntax
```
HWND GetDlgItem(HWND Dlg, int IDDlgItem);
```

Beschreibung Diese Funktion liefert das Handle eines im Dialogfenster enthaltenen Steuerelements. Das Dialogfenster wird durch den Parameter *Dlg* bestimmt.

Dlg: Bezeichnet das Dialogfenster, in dem das Steuerelement enthalten ist.

IDDlgItem: Bestimmt die Integer-ID des zu ermittelnden Elements.

Rückgabewert Der Rückgabewert bezeichnet das vorgegebene Steuerelement. Er ist NULL, wenn kein Steuerelement mit der durch *IDDlgItem* angegebenen Integer-ID existiert.

Hinweis Die Funktion *GetDlgItem* kann für jedes Paar über- und untergeordneter Fenster verwendet werden, sie ist also nicht nur bei Dialogfenstern anwendbar. Wenn *Dlg* ein übergeordnetes Fenster bezeichnet und das untergeordnete Fenster eine eindeutige ID besitzt, liefert *GetDlgItem* ein gültiges Handle für das untergeordnete Fenster.

Die ID des untergeordneten Fensters wurde durch den Parameter *Menu* der erzeugenden Funktion *CreateWindow* festgelegt.

GetDlgItemInt

Syntax
```
WORD GetDlgItemInt(HWND Dlg, int IDDlgItem,
BOOL FAR * Translate, BOOL Signed);
```

Beschreibung Diese Funktion übersetzt den Text eines Steuerelements des gegebenen Dialogfensters in einen Integer. *GetDlgItemInt* ermittelt den Text des durch den Parameter *IDDlgItem* bezeichneten Steuerelements und übersetzt diesen ohne Berücksichtigung etwaiger Leerzeichen am Anfang des Textes durch Konvertierung der Dezimalziffern. Die Übersetzung wird beendet, wenn das Ende des Textes erreicht ist oder ein nichtnumerisches Zeichen aufgefunden wird.

Ist der Parameter *Signed* ungleich Null, prüft *GetDlgItemInt* auf ein Minuszeichen (-) am Anfang des Textes. Ist dieses vorhanden, so wird der Text in eine vorzeichenbehaftete Zahl übersetzt, andernfalls ist die Zahl vorzeichenlos.

GetDlgItemInt gibt 0 zurück, wenn die übersetzte Zahl bei negativen Werten absolut größer als 32767 und bei positiven Werten größer als 65535 ist. Treten Fehler auf, wie z.B. das Auffinden eines nichtnumerischen Zeichens oder das Überschreiten des vorgegebenen Höchstwerts, kopiert die Funktion Null in das Ziel, auf das der Parameter *Translate* zeigt. Treten keine Fehler auf, erhält der Parameter *Translate* einen Wert ungleich Null. Ist *Translate* Null, werden keine Warnungen ausgegeben.

Die Funktion übergibt eine *WM_GETTEXT*-Botschaft an das Steuerelement.

Dlg: Bezeichnet das Dialogfenster.

IDDlgItem: Bestimmt den Integer-Bezeichner des zu übersetzenden Objekts des Dialogfensters.

Translate: Zurückgegebener Bool-Wert. Er ist Null, wenn bei der Übersetzung ein Fehler auftritt.

Signed: Bestimmt, ob der zu ermittelnde Wert vorzeichenbehaftet ist.

Rückgabewert Der Rückgabewert gibt den übersetzten Wert des Textes des Dialogfensterelements an. Da Null ein gültiger Rückgabewert ist, muß der Parameter *Translate* zum Auffinden von Fehlern verwendet werden.

Querverweis *WM_GETTEXT*

GetDlgItemText

Syntax
```
int GetDlgItemText(HWND Dlg, int IDDlgItem, LPSTR Str,
int MaxCount);
```

Beschreibung Diese Funktion ermittelt den zu einem Steuerelement in einem Dialogfenster gehörenden Text bzw. dessen Überschrift und kopiert den Text an die durch den Parameter *Str* bezeichnete Position. Anschließend wird eine Zahl zurückgegeben, die der Anzahl der kopierten Zeichen entspricht.

GetDlgItemText übergibt eine *WM_GETTEXT*-Botschaft an das Steuerelement.

Dlg: Bezeichnet das Dialogfenster, in dem sich das gewünschte Steuerelement befindet.

IDDlgItem: Bestimmt den Integer-Bezeichner des Objekts im Dialogfenster, dessen Überschrift oder Text ermittelt werden soll.

Str: Zeigt auf den Puffer, der den Text aufnimmt.

MaxCount: Bestimmt die maximale Länge des in *Str* zu kopierenden Strings in Bytes. Ist der String länger als in *MaxCount* angegeben, werden die überzähligen Zeichen abgeschnitten.

Rückgabewert Der Rückgabewert gibt die Anzahl der tatsächlich in den Puffer kopierten Zeichen an. Er ist 0, wenn kein Text kopiert wurde.

Querverweis *WM_GETTEXT*

GetDOSEnvironment

Syntax
```
LPSTR GetDOSEnvironment(void);
```

Beschreibung Diese Funktion gibt einen Zeiger auf den Environment-String der laufenden Task zurück. Weitere Informationen über Format und Inhalt des Environment-Strings finden sich in der *Microsoft MS-DOS Programmer's Reference*.

Die Funktion hat keine Parameter.

Rückgabewert Diese Funktion gibt einen Zeiger auf den Environment-String der laufenden Task zurück.

Hinweis Im Unterschied zu einer normalen Anwendung besitzt eine DLL (dynamische Linkbibliothek) keine Kopie des Environment-Strings. Aus diesem Grund muß sie *GetDOSEnvironment* zur Ermittlung des Strings aufrufen.

GetDoubleClickTime

Syntax
```
WORD GetDoubleClickTime(void);
```

Beschreibung Diese Funktion ermittelt die Zeit, die für einen Doppelklick der Maus zur Verfügung steht. Ein Doppelklick ist die kurz aufeinanderfolgende, zweimalige Betätigung der Maustaste, wobei die zweite Betätigung innerhalb einer festgelegten Zeit erfol-

gen muß. Die Doppelklickzeit ist die Anzahl von Millisekunden, die maximal zwischen zwei Tastenbetätigungen verstreichen darf.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert gibt die aktuelle Doppelklickzeit in Millisekunden an.

GetDriveType

Syntax
```
WORD GetDriveType(int Drive);
```

Beschreibung Diese Funktion stellt fest, ob in einem Laufwerk austauschbare oder fest installierte Speichermedien verwendet werden oder ob es sich um ein Netzlaufwerk handelt.

Drive: Bestimmt das Laufwerk, dessen Typ ermittelt werden soll. Laufwerk A: hat den Wert 0, B: den Wert 1, Laufwerk C: den Wert 2 usw.

Rückgabewert Der Rückgabewert gibt den Laufwerktyp an. Möglich sind folgende Werte:

Wert	Bedeutung
DRIVE_REMOVEABLE	Diskettenlaufwerk.
DRIVE_FIXED	Festplattenlaufwerk.
DRIVE_REMOTE	Netzlaufwerk.

Der Rückgabewert ist 0, wenn die Funktion den Laufwerktyp nicht ermitteln kann, und 1, wenn das angegebene Laufwerk nicht existiert.

GetEnvironment

Syntax
```
int GetEnvironment(LPSTR PortName, LPSTR Environ, WORD
MaxCount);
```

Beschreibung Diese Funktion ermittelt die aktuelle Umgebung der an eine durch den Parameter *PortName* bezeichneten Systemschnittstelle angeschlossenen Geräteeinheit. Die Information wird in den Puffer kopiert, auf den der Parameter *Environ* zeigt.

Die von der grafischen Geräteschnittstelle (GDI) unterstützte Umgebung enthält binäre Daten, die von der GDI zur Erstellung eines Gerätekontexts für die Geräteeinheit an der gegebenen Schnittstelle verwendet werden.

Die Funktion führt nicht zum gewünschten Ergebnis, wenn keine Umgebung der vorgegebenen Schnittstelle aufgefunden werden kann.

Anwendungen können die Funktion mit dem auf NULL

gesetzten Parameter *Environ* aufrufen, um die nötige Größe des Puffers für die Umgebung festzustellen. Anschließend kann für den Puffer Speicher reserviert und ein weiteres Mal die Funktion *GetEnvironment* aufgerufen werden, um die Umgebung zu ermitteln.

PortName: Zeigt auf den nullterminierten String, der den Namen der gewünschten Schnittstelle enthält.

Environ: Zeigt auf den Puffer, der die Umgebung aufnehmen soll.

MaxCount: Bestimmt die maximale Anzahl der in den Puffer zu kopierenden Bytes.

Rückgabewert Der Rückgabewert ist die Anzahl der in *Environ* kopierten Bytes. Ist der Parameter *Environ* auf NULL gesetzt, wird die zum Speichern der Umgebung benötigte Größe des Puffers zurückgegeben. Kann die Umgebung nicht gefunden werden, ist der Rückgabewert 0.

Hinweis Das erste Feld des Puffers, auf das der Parameter *Environ* zeigt, muß identisch mit dem im Parameter *DeviceName* an die Funktion *CreateDC* übergebenen Wert sein.

Bestimmt *PortName* eine Null-Schnittstelle (wie sie in der Datei WIN.INI definiert ist), wird der durch *Environ* bezeichnete Gerätename verwendet, um die gewünschte Umgebung aufzufinden.

GetFocus

Syntax
```
HWND GetFocus(void);
```

Beschreibung Diese Funktion ermittelt das Handle des Fensters, das aktuell den Fokus besitzt.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bezeichnet das Fenster, das aktuell den Fokus besitzt. Konnte die Funktion nicht erfolgreich ausgeführt werden, ist der Rückgabewert NULL.

GetFreeSpace

Syntax
```
DWORD GetFreeSpace(WORD Flag);
```

Beschreibung Diese Funktion untersucht den globalen Heap und gibt den zur Verfügung stehenden Speicher in Bytes zurück.

Flag: Bestimmt, ob der Heap über oder unter der EMS-Speicherabgrenzung in Large-Frame- und Small-Frame-EMS-Systemen geprüft wird. Ist der Parameter auf *GMEM_NOT_BANKED* gesetzt, gibt *GetFreeSpace* die Größe des verfügbaren Speichers

unterhalb der Abgrenzung zurück. Der Wert oberhalb der Abgrenzung wird zurückgegeben, wenn *Flag* den Wert 0 hat. In Systemen ohne EMS wird dieser Parameter ignoriert.

Rückgabewert Der Rückgabewert entspricht der Größe des verfügbaren Speichers in Byte. Dieser Speicher muß nicht notwendigerweise zusammenhängend sein.

Die Funktion *GlobalCompact* ermittelt die Anzahl der Bytes des größten zusammenhängenden Blocks im globalen Speicher.

Querverweis *GlobalCompact*

GetGValue

Syntax
```
BYTE GetGValue(DWORD RGBColor);
```

Beschreibung Dieses Makro ermittelt den grünen Farbanteil eines RGB-Farbwerts.

RGBColor: Bestimmt ein rotes, ein grünes und ein blaues Farbfeld. Diese Felder bestimmen die Intensität der gegebenen Farbe.

Rückgabewert Der Rückgabewert ist ein Byte, das den grünen RGB-Farbwert enthält.

Hinweis Der Wert 0x0FF entspricht der maximalen Intensität eines einzelnen Bytes, 0x000 entspricht der minimalen Intensität eines einzelnen Bytes.

GetInputState

Syntax
```
BOOL GetInputState(void);
```

Beschreibung Diese Funktion untersucht, ob in der System-Warteschlange Maus-, Tastatur- oder Timer-Ereignisse zur Bearbeitung vorliegen.

Ein Ereignis ist ein Record, der Eingaben auf der Interrupt-Ebene beschreibt. Mausereignisse liegen bei Mausbewegungen oder Mausklicks vor. Ein Tastaturereignis ist jede Betätigung einer oder mehrere Tasten. Timer-Ereignisse liegen nach einer vorgegebenen Anzahl von Clock-Ticks vor. Solche Ereignisse werden von Windows in der System-Warteschlange gespeichert.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bestimmt, ob Maus-, Tastatur- oder Timer-Ereignisse vorliegen. Ist dies nicht der Fall, ist der Rückgabewert Null.

GetInstanceData

Syntax
```
int GetInstanceData(HANDLE Instance, NPSTR Data,
int Count);
```

Beschreibung Diese Funktion kopiert Daten einer vorhergehenden Instanz der Anwendung in den Datenbereich der aktuellen Instanz. Der Parameter *Instance* legt fest, aus welcher Instanz die Daten in den durch *Data* bezeichneten Bereich kopiert werden sollen. Der Parameter *Count* bestimmt die Anzahl der zu kopierenden Bytes.

Instance: Bezeichnet eine vorhergehende Instanz der Anwendung.

Data: Zeigt auf einen Puffer in der aktuellen Instanz.

Count: Bestimmt die Anzahl der zu kopierenden Bytes.

Rückgabewert Der Rückgabewert entspricht der Anzahl der tatsächlich kopierten Bytes.

GetKBCodePage

Syntax
```
int GetKBCodePage();
```

Beschreibung Diese Funktion ermittelt, welche OEM/ANSI-Tabelle von Windows geladen ist.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert gibt die gerade von Windows geladene Code-Seite an.
Möglich sind die in folgender Tabelle aufgeführten Werte:

Wert	Bedeutung
437	Standard (USA, wird in den meisten Ländern verwendet. Zeigt an, daß sich keine Datei OEMANSI.BIN im Windows-Verzeichnis befindet).
850	International (OEMANSI.BIN = XLAT850.BIN)
860	Portugal (OEMANSI.BIN = XLAT860.BIN)
861	Island (OEMANSI.BIN = XLAT861.BIN)
863	Franko-Kanadisch (OEMANSI.BIN = XLAT863.BIN)
865	Norwegen/Dänemark (OEMANSI.BIN = XLAT865.BIN)

Hinweis Befindet sich die Datei OEMANSI.BIN im Windows-Verzeichnis, wird sie von Windows gelesen. Die OEM/ANSI-Übersetzungstabellen im Tastaturtreiber werden von ihr überschrieben.

Wählt der Anwender innerhalb des Setup-Programms eine Sprache, die nicht die Standard-Code-Seite (437) benutzt,

kopiert Setup die passende Datei (zum Beispiel XLATPO.BIN) in OEMANSI.BIN im Windows-Systemverzeichnis.

Wird die Standard-Code-Seite verwendet, löscht Setup die Datei OEMANSI.BIN aus dem Windows-Systemverzeichnis.

GetKeyboardState

Syntax
```
void GetKeyboardState(BYTE FAR * KeyState);
```

Beschreibung Diese Funktion kopiert den Status der 256 möglichen Tasten der virtuellen Tastatur in den durch *KeyState* bezeichneten Puffer. Das obere Bit eines jeden Bytes wird auf 1 gesetzt, wenn die Taste gedrückt ist, auf 0, wenn die Taste nicht betätigt wird. Das untere Bit wird auf 1 gesetzt, wenn die Taste seit dem Startup betätigt wurde, andernfalls enthält es den Wert 0.

KeyState: Ein 256-Byte-Puffer für die virtuellen Tasten-Codes.

Rückgabewert Keiner.

Hinweis Eine Anwendung ruft die Funktion *GetKeyboardState* als Reaktion auf eine Tatstatureingabebotschaft auf. Diese Funktion ermittelt den Zustand der Tastatur bei Erstellung der Botschaft.

Um Informationen über einzelne Tasten zu erhalten, müssen folgende Schritte durchgeführt werden:

1. Erzeugen eines Zeichen-Arrays mit 256 Byte Länge.

2. Kopieren des durch *KeyState* bezeichneten Pufferinhalts in dieses Array.

3. Überprüfung der virtuellen Tasten-Codes, um den Status einzelner Tasten zu ermitteln.

GetKeyboardType

Syntax
```
int GetKeyboardType(int TypeFlag);
```

Beschreibung Diese Funktion ermittelt den Typ der vom System verwendeten Tastatur.

TypeFlag: Bestimmt, ob der Rückgabewert der Funktion den Typ oder den Untertyp der Tastatur anzeigt. Möglich sind die in der folgenden Tabelle aufgeführten Werte:

Wert	Bedeutung
0	Die Funktion gibt den Typ der Tastatur zurück.
1	Die Funktion gibt den Untertyp der Tastatur zurück.
2	Die Funktion gibt die Anzahl der Funktionstasten der Tastatur zurück.

Rückgabewert Der Rückgabewert zeigt den Typ oder Untertyp der vom System verwendeten Tastatur oder die Anzahl der existierenden Funktionstasten an. Der Untertyp ist ein OEM-abhängiger Wert. Folgende Werte sind möglich:

Wert	Bedeutung
1	IBM PC/XT oder kompatible Tastatur (83 Tasten)
2	Olivetti M24 "ICO" Tastatur (102 Tasten)
3	IBM AT oder vergleichbare Tastatur (84 Tasten)
4	IBM Erweiterte Tastatur (101 oder 102 Tasten)
5	Nokia 1050 und vergleichbare Tastaturen
6	Nokia 9140 und vergleichbare Tastaturen

Der Rückgabewert ist Null, wenn die Funktion fehlschlägt oder der Parameter *TypeFlag* einen Wert größer 2 hat.

Hinweis Eine Anwendung kann die Anzahl der Funktionstasten auf der Tastatur über deren Typ ermitteln.

Die folgende Tabelle zeigt die Anzahl der Funktionstasten jedes Tastaturtyps:

Typ	Anzahl der Funktionstasten
1	10
2	12 (oder 18)
3	10
4	12
5	10
6	24

GetKeyNameText

Syntax
```
int GetKeyNameText(LONG lParam, LPSTR Buffer, int Size);
```

Beschreibung Diese Funktion ermittelt einen String, der den Namen einer Taste enthält.

Der Tastaturtreiber enthält eine Liste von Namen in Form von Zeichenketten für die Tasten, deren Namen länger als ein einzelnes Zeichen sind. Der Tastenname wird entsprechend der Tastenanordnung der installierten Tastatur übersetzt. Die Übersetzung wird entsprechend der durch den Tastaturtreiber unterstützten Sprache vorgenommen.

lParam: Bestimmt den 32-Bit-Parameter der Tastaturbotschaft, die die Funktion gerade bearbeitet (z.B. *WM_KEYDOWN*). Byte 3 (Bits 16 bis 23) dieses Parameters ist ein Scan-Code. Bit 20 ist ein zusätzliches Bit, das einige Tasten auf einer erweiterten Tastatur unterscheidet. Bit 21 wird von der Anwendung gesetzt, um die Funktion anzuweisen, nicht zwischen linken

oder rechten Ctrl- bzw. Shift-Tasten (etc.) zu unterscheiden. Beispiel: 0x10FF00 bestimmt die Ctrl-Taste.

Buffer: Zeigt auf den nullterminierten String, der den Tastennamen aufnehmen soll (auf »STRG« im o. a. Beispiel).

Size: Bestimmt die maximale Länge des Tastennamens in Byte. In diesen Wert ist die abschließende 0 nicht eingeschlossen.

Rückgabewert Der Rückgabewert entspricht der tatsächlichen Länge des in *Buffer* kopierten Strings (ohne abschließende 0).

GetKeyState

Syntax
```
int GetKeyState(int VirtKey);
```

Beschreibung Diese Funktion ermittelt den Status der durch den Parameter *VirtKey* bestimmten virtuellen Taste. Der Status zeigt an, ob die Taste gedrückt oder arretiert ist.

VirtKey: Bestimmt eine virtuelle Taste. Handelt es sich bei der angeforderten Taste um einen Buchstaben oder um eine Ziffer (A bis Z, a bis z, 0 bis 9), muß *VirtKey* der ASCII-Wert des Zeichens zugewiesen werden. Für die anderen Tasten muß ein gültiger virtueller Tasten-Code verwendet werden.

Rückgabewert Der Rückgabewert gibt den Status der vorgegebenen virtuellen Taste an. Ist das höherwertige Bit 1, wird die Taste gedrückt. Ist das niederwertige Bit 1, ist die Taste arretiert. Eine Arretierungstaste wie CapsLock ist arretiert, wenn sie seit dem Systemstart eine ungerade Anzahl mal betätigt wurde. Ist das Bit 0, ist die Taste nicht arretiert.

Hinweis Eine Anwendung ruft *GetKeyState* als Reaktion auf eine Tastatureingabebotschaft auf.

Die Funktion ermittelt den Status der Taste während der Erzeugung der Botschaft.

GetLastActivePopup

Syntax
```
HWND GetLastActivePopup(HWND WndOwner);
```

Beschreibung Diese Funktion ermittelt, welches zu dem durch *WndOwner* bezeichneten Fenster gehörige Pop-up-Fenster zuletzt aktiviert war.

WndOwner: Bezeichnet das besitzende Fenster.

Rückgabewert Der Rückgabewert bezeichnet das zuletzt aktivierte Pop-up-Fenster.

Der Rückgabewert ist identisch mit dem Parameter *WndOwner*, wenn eine der folgenden Bedingungen zutrifft:

- Das durch *WndOwner* bezeichnete Fenster war selbst das zuletzt aktivierte.
- Das durch *WndOwner* bezeichnete Fenster hat keine Pop-up-Fenster.
- Das durch *WndOwner* bezeichnete Fenster ist kein Hauptfenster oder es gehört zu einem anderen Fenster.

GetMapMode

Syntax `int GetMapMode(HDC DC);`

Beschreibung Diese Funktion liefert den aktuellen Abbildungsmodus. Die Beschreibung der möglichen Werte findet sich in Kapitel 4 »Windows-Stile und -Konstanten« in Abschnitt »MM_XXXX«.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert bestimmt den Abbildungsmodus.

Querverweis *SetMapMode*

GetMenu

Syntax `HMENU GetMenu(HWND Wnd);`

Beschreibung Liefert das Handle eines Menüs des gegebenen Fensters.

Wnd: Bezeichnet das Fenster, dessen Menü untersucht werden soll.

Rückgabewert Der Rückgabewert bezeichnet das Menü. Er ist NULL, wenn das Fenster kein Menü besitzt. Wenn das Fenster untergeordnet ist, ist der Rückgabewert nicht definiert.

GetMenuCheckMarkDimensions

Syntax `LONG GetMenuCheckMarkDimensions(void);`

Beschreibung Diese Funktion gibt die Größe des Bitmaps der Standardauswahlmarkierung (Häkchen) zurück. Windows stellt dieses Bitmap neben den markierten Menüeinträgen dar. Vor der Ersetzung des Häkchens durch die Funktion *SetMenuItemBitmaps* sollte die Anwendung die korrekte Größe der Bitmaps mit *GetMenuCheckMarkDimensions* ermitteln.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bestimmt Höhe und Breite des Bitmaps der Standardauswahlmarkierung. Das höherwertige WORD enthält die Höhe, das niederwertige die Breite in Pixel.

Querverweis *SetMenuItemBitmaps*

GetMenuItemCount

Syntax
```
WORD GetMenuItemCount(HMENU Menu);
```

Beschreibung Diese Funktion stellt die Anzahl der Einträge in dem durch *Menu* bezeichneten Menü fest. Es kann sich um ein Pop-up-Menü oder ein Hauptmenü handeln.

Menu: Bezeichnet das Handle des zu untersuchenden Menüs.

Rückgabewert Der Rückgabewert gibt die Anzahl der Menüeinträge im durch *Menu* bezeichneten Menü an. Wenn die Funktion nicht erfolgreich ausgeführt werden konnte, ist der Rückgabewert -1.

GetMenuItemID

Syntax
```
WORD GetMenuItemID(HMENU Menu, int Pos);
```

Beschreibung Diese Funktion ermittelt den Bezeichner des Menüeintrags, der sich an der durch *Pos* bezeichneten Position befindet.

Menu: Bezeichnet das Handle des Pop-up-Menüs, das den Eintrag enthält, dessen Bezeichner ermittelt wird.

Pos: Gibt die Position des zum Bezeichner gehörenden Menüeintrags (ausgehend von 0) an.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt wurde, gibt der Rückgabewert die ID eines Eintrags in einem Pop-up-Menü an. Hat der Parameter *Menu* den Wert 0 oder ist der betreffende Eintrag selbst ein Pop-up-Menü, ist der Rückgabewert -1.

GetMenuState

Syntax
```
WORD GetMenuState(HMENU Menu, WORD ID, WORD Flags);
```

Beschreibung Diese Funktion ermittelt die Anzahl der Einträge im Pop-up-Menü, das mit dem durch *ID* bezeichneten Menüeintrag verbunden ist.

Voraussetzung ist, daß der Parameter *Menu* ein zu einem Pop-up-Menü gehörendes Menü bezeichnet. Bezeichnet *Menu* ein Pop-up-Menü, ermittelt die Funktion den Status des durch *ID* bezeichneten Menüeintrags.

Menu: Bezeichnet das Menü.

ID: Bestimmt die *ID* des Menüeintrags.

Flags: Bestimmt die Interpretation des Parameters *ID*. Enthält der Parameter *Flags MF_BYPOSITION*, legt *ID* eine von 0 ausgehende, relative Position fest. Enthält *Flags MF_BYCOMMAND*, bestimmt *ID* die ID des Eintrags.

Rückgabewert Der Rückgabewert ist -1, wenn der angegebene Eintrag nicht existiert. Existiert auch das Menü nicht, wird ein Fatal Exit (nicht behebbarer Fehler) im Anwendungsprogramm ausgelöst.

Bezeichnet *ID* ein Pop-up-Menü, enthält der Rückgabewert die Anzahl der Menüeinträge im höherwertigen Byte und die zum Menü gehörenden Flags im niederwertigen Byte. Andernfalls liegt eine Maske mit dem logischen ODER (&&) verknüpfter Werte aus folgender Liste vor. Diese Maske beschreibt den Status des durch *ID* bestimmten Menüeintrags:

Wert	Bedeutung
MF_CHECKED	Auswahlmarkierung neben dem Eintrag (nur bei Pop-up-Menüs).
MF_DISABLED	Eintrag ist deaktiviert.
MF_ENABLED	Eintrag kann angewählt werden.
MF_GRAYED	Eintrag ist in einen nicht aktivierbaren Zustand versetzt und grau hinterlegt.
MF_MENUBARBREAK	Wie *MF_MENUBREAK*, nur wird bei Pop-up-Menüs die neue Spalte durch eine vertikale Linie von der alten getrennt.
MF_MENUBREAK	Eintrag ist in einer neuen Zeile (statisches Menü) oder einer neuen Spalte (Pop-up-Menü) ohne Trennung der Spalten positioniert.
MF_SEPARATOR	Horizontale Trennlinie wird gezeichnet (nur bei Pop-up-Menüs). Diese Linie kann nicht angewählt, markiert, in nicht aktivierbaren Zustand versetzt oder hervorgehoben werden.
MF_UNCHECKED	Keine Auswahlmarkierung neben dem Eintrag (Standardeinstellung).

GetMenuString

Syntax
```
int GetMenuString(HMENU Menu, WORD IDItem, LPSTR Str,
int MaxCount, WORD Flag);
```

Beschreibung Diese Funktion kopiert das Label des angegebenen Menüeintrags in den Parameter *Str*.

Menu: Bezeichnet das Menü.

IDItem: Bestimmt den Bezeichner (Integer) des Menüeintrags aus der Ressourcendatei oder den Offset des Menüeintrags im Menü in Abhängigkeit vom Wert des Parameters *Flag*.

Str: Zeigt auf den nullterminierten String, der das Label aufnimmt.

MaxCount: Bestimmt die maximale Länge des zu kopierenden Labels. Ist das Label länger, werden die überzähligen Zeichen abgeschnitten.

Flag: Bestimmt den Typ des Parameters *IDItem*. Enthält *Flag MF_BYPOSITION*, gibt *IDItem* eine von 0 ausgehende, relative Position an, enthält *Flag MF_BYCOMMAND*, gibt der Parameter *IDItem* die ID des Eintrags an.

Rückgabewert Der Rückgabewert entspricht der Anzahl der tatsächlich in den Puffer kopierten Bytes.

Hinweis Der Parameter *MaxCount* sollte die Anzahl der Zeichen im Label um 1 überschreiten, damit die den String abschließende Null aufgenommen werden kann.

GetMessage

Syntax
```
BOOL GetMessage(LPMSG Msg, HWND Wnd, WORD MsgFilterMin,
WORD MsgFilterMax);
```

Beschreibung Diese Funktion liefert eine Botschaft von der Anwendungs-Warteschlange und legt diese Botschaft in der *MSG*-Struktur *Msg* ab.

Ist keine Botschaft verfügbar, übergibt *GetMessage* die Steuerung an andere Anwendungen, bis eine Botschaft verfügbar wird.

GetMessage liefert nur Botschaften, die dem durch den Parameter *Wnd* bezeichneten Fenster zugeordnet sind und die innerhalb des durch die Parameter *MsgFilterMin* und *MsgFilterMax* vorgegebenen Bereichs von Botschaftswerten liegen. Wenn *Wnd* 0 ist, liefert *GetMessage* Botschaften für jedes Fenster, das zur aufrufenden Anwendung gehört. (Die Funktion *GetMessage* liefert keine Botschaften für Fenster, die zu einer anderen Anwendung gehören). Sind *MsgFilterMin* und *MsgFilterMax* beide 0, gibt *GetMessage* alle verfügbaren Botschaften zurück (es wird keine Filterung durchgeführt).

Die Konstanten *WM_KEYFIRST* und *WM_KEYLAST* können als Filterwerte benutzt werden, um alle Botschaften abzurufen, die sich auf Tastatureingaben beziehen.

Die Konstanten *WM_MOUSEFIRST* und *WM_MOUSELAST* können benutzt werden, um alle mausbezogenen Botschaften abzurufen.

Msg: Zeigt auf eine *MSG*-Datenstruktur, die Botschaftsinformationen aus der Windows-Anwendungswarteschlange enthält.

Wnd: Bezeichnet das Fenster, dessen Botschaften verarbeitet werden sollen. Wenn *Wnd* 0 ist, liefert *GetMessage* Botschaften für jedes Fenster, das zur aufrufenden Anwendung gehört.

MsgFilterMin: Bestimmt den Wert des niedrigsten abzurufenden Botschaftswerts.

MsgFilterMax: Bestimmt den Wert des höchsten abzurufenden Botschaftswerts.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn eine andere Botschaft als *WM_QUIT* abgerufen wurde, er ist Null, wenn die Botschaft *WM_QUIT* abgerufen wurde.

Der Rückgabewert wird normalerweise benutzt, um zu entscheiden, ob die Hauptschleife der Anwendung beendet und das Programm verlassen werden soll.

Hinweis Die Funktionen *PeekMessage* und *GetMessage* geben die Steuerung an andere Anwendungen weiter, wenn Botschaften *WM_- PAINT* oder *WM_TIMER* für andere Tasks vorliegen oder wenn keine Botschaften verfügbar sind. Ein Aufruf einer der Funktionen *GetMessage*, *PeekMessage* und *WaitMessage* ist die einzige Möglichkeit, andere Anwendungen laufen zu lassen. Solange eine Anwendung keine dieser Funktionen aufruft, können keine anderen Anwendungen laufen.

Geben *GetMessage*, *PeekMessage* und *WaitMessage* die Steuerung an andere Anwendungen ab, können sich die Stack- und Datensegmente der die Funktion aufrufenden Anwendung verschieben, um sich dem veränderten Speicherplatzbedarf anderer Anwendungen anzupassen. Wenn die Anwendung Zeiger auf Objekte im Daten- oder Stacksegment (das sind globale bzw. lokale Variablen) gespeichert hat, können diese Zeiger nach einem Aufruf von *GetMessage*, *PeekMessage* oder *WaitMessage* ungültig werden. Der Parameter *Msg* der aufgerufenen Funktion bleibt in jedem Fall gültig.

Querverweis *PeekMessage*, *WaitMessage*

GetMessagePos

Syntax
```
DWORD GetMessagePos(void);
```

Beschreibung Diese Funktion gibt einen Wert zurück, der die Mauszeigerposition in Bildschirmkoordinaten zu dem Zeitpunkt darstellt, zu dem die letzte von der Funktion *GetMessage* abgerufene Botschaft eintraf.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert gibt die x- und y-Koordinaten der Mauszeigerposition zurück. Die x-Koordinate liegt im niederwertigen, die y-Koordinate im höherwertigen WORD vor. Wenn der Rückgabewert einer Variablen zugewiesen ist, kann der Typbezeichner *MakePoint* benutzt werden, um eine *POINT*-Struktur aus dem Rückgabewert zu erhalten.

Hinweis Um die aktuelle Position des Mauszeigers anstelle seiner Position beim letzten Eintreffen der Botschaft zu erhalten, kann die Funktion *GetCursorPos* verwendet werden.

GetMessageTime

Syntax
```
LONG GetMessageTime(void);
```

Beschreibung Diese Funktion gibt die Zeit der Erzeugung der letzten von *GetMessage* abgerufenen Botschaft an. Sie bezeichnet die Zeit in Millisekunden, die vom Booten des Systems bis zu dem Zeitpunkt, an dem die Botschaft erzeugt (in die Warteschlange gestellt) wurde, vergangen ist.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert gibt die Zeit der Erzeugung der Botschaft zurück.

Hinweis Man sollte nicht voraussetzen, daß sich der Rückgabewert immer vergrößert. Der Rückgabewert wird auf 0 zurückgesetzt, wenn der Timer-Zählwert den Maximalwert für LONG-Werte überschreitet.

Um das Zeitintervall zwischen Botschaften zu berechnen, subtrahiert man die Zeit der ersten Botschaft von der der zweiten.

GetMetaFile

Syntax
```
HANDLE GetMetaFile(LPSTR FileName);
```

Beschreibung Diese Funktion erzeugt ein Handle für das Metafile, das vom Parameter *FileName* benannt wird.

FileName: Zeigt auf den nullterminierten String, der den DOS-Dateinamen des Metafiles enthält. Es wird vorausgesetzt, daß das Metafile existiert.

Rückgabewert Der Rückgabewert bezeichnet ein Metafile, wenn die Funktion erfolgreich war, andernfalls ist er 0.

GetMetaFileBits

Syntax
```
HANDLE GetMetaFileBits(HANDLE MF);
```

Beschreibung Diese Funktion gibt ein Handle auf einen globalen Speicherblock zurück, der das angegebene Metafile als eine Ansammlung von Bits enthält. Der Speicherblock kann verwendet werden, um die Größe des Metafiles zu bestimmen oder um das Metafile als Datei abzuspeichern. Der Speicherblock sollte nicht verändert werden.

MF: Bezeichnet das Speicher-Metafile.

Rückgabewert Der Rückgabewert bezeichnet den globalen Speicherblock, der das Metafile enthält. Wenn ein Fehler auftritt, ist der Rückgabewert 0.

Hinweis Das als Parameter *MF* verwendete Handle wird ungültig, wenn die Funktion *GetMetaFileBits* zurückkehrt. Aus diesem Grund muß das zurückgegebene globale Speicher-Handle benutzt werden, um sich auf das Metafile zu beziehen.

Von dieser Funktion erzeugte Speicherblöcke sind nur für die aufrufende Anwendung bestimmt und werden von anderen Anwendungen nicht verwendet. Diese Blöcke werden automatisch gelöscht, wenn die Anwendung beendet wird.

GetModuleFileName

Syntax
```
int GetModuleFileName(HANDLE Module, LPSTR FileName,
int Size);
```

Beschreibung Diese Funktion liefert den vollständigen Pfadnamen (Laufwerk, Verzeichnis, Name, Extension) der ausführbaren Datei, aus der das angegebene Modul geladen wurde.

GetModuleFileName kopiert den Dateinamen (nullterminiert) in den Puffer, auf den der Parameter *FileName* zeigt.

Module: Bezeichnet das Modul oder die Instanz des Moduls.

FileName: Zeigt auf den nullterminierten String, der den Dateinamen aufnehmen soll.

Size: Bestimmt die maximale Anzahl von Zeichen, die kopiert werden sollen. Wenn der Dateiname länger als die vom Parameter *Size* angegebene Maximalzahl von Zeichen ist, wird er abgeschnitten.

Rückgabewert Der Rückgabewert gibt die tatsächliche Länge des in den Puffer kopierten Strings an.

GetModuleHandle

Syntax
```
HANDLE GetModuleHandle(LPSTR ModuleName);
```

Beschreibung Diese Funktion liefert das Modul-Handle des angegebenen Moduls.

ModuleName: Zeigt auf einen nullterminierten String, der das Modul bezeichnet.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt werden konnte, bezeichnet der Rückgabewert das Modul, andernfalls wird 0 zurückgegeben.

GetModuleUsage

Syntax
```
int GetModuleUsage(HANDLE Module);
```

Beschreibung Diese Funktion liefert den Referenzzähler eines angegebenen Moduls.

Module: Bezeichnet das Modul oder die Instanz des Moduls.

Rückgabewert Der Rückgabewert bezeichnet den Referenzzähler des Moduls.

GetNearestColor

Syntax
```
DWORD GetNearestColor(HDC DC, COLORREF Color);
```

Beschreibung Diese Funktion gibt die einer angegebenen virtuellen Farbe ähnlichste virtuelle Farbe zurück, die vom angegebene Gerät dargestellt werden kann.

DC: Bezeichnet den Gerätekontext.

Color: Bestimmt die Farbe, die angeglichen werden soll.

Rückgabewert Der Rückgabewert zeigt einen RGB-Farbwert an, der die Flächenfarbe bezeichnet, die das Gerät darstellen kann und der dem Wert *Color* am nächsten kommt.

GetNearestPaletteIndex

Syntax
```
WORD GetNearestPaletteIndex(HPALETTE Palette,
COLORREF Color);
```

Beschreibung Diese Funktion gibt den Index des Eintrags in einer virtuellen Palette zurück, der einem vorgegebenen Farbwert am ähnlichsten ist.

Palette: Bezeichnet die virtuelle Palette.

Color: Bestimmt die Farbe, die angeglichen werden soll.

Rückgabewert Der Rückgabewert ist der Index des Eintrags in einer virtuellen Palette. Der Eintrag enthält die Farbe, die der angegebenen Farbe am ähnlichsten ist.

GetNextDlgGroupItem

Syntax
```
HWND GetNextDlgGroupItem(HWND Dlg, HWND Ctrl,
BOOL Previous);
```

Beschreibung Diese Funktion sucht in dem vom Parameter *Dlg* bezeichneten Dialogfenster nach dem nächsten (oder vorherigen) Steuerelement innerhalb einer Gruppe von Steuerelementen. Eine

Gruppe von Steuerelementen besteht aus einer oder mehreren Steuerelementen mit dem Stil *WS_GROUP*.

Dlg: Bezeichnet das zu durchsuchende Dialogfenster.

Ctrl: Bezeichnet das Steuerelement im Dialogfenster, bei dem die Suche beginnt.

Previous: Bestimmt, wie die Funktion die Gruppe von Steuerungen im Dialogfenster absuchen soll. Ist *Previous* Null, sucht die Funktion nach dem vorhergehenden Steuerelement in der Gruppe. Ist *Previous* ungleich Null, sucht die Funktion nach dem nächsten Steuerelement in der Gruppe.

Rückgabewert Der Rückgabewert bezeichnet das nächste oder vorherige Steuerelement in der Gruppe.

Hinweis Ist das aktuelle Element das letzte Element in der Gruppe und ist *Previous* Null, gibt die Funktion *GetNextDlgGroupItem* das Fenster-Handle des ersten Elements in der Gruppe zurück. Ist das aktuelle Element das erste Element in der Gruppe und ist *Previous* ungleich Null, gibt die Funktion *GetNextDlgGroupItem* das Fenster-Handle des letzten Elements in der Gruppe zurück.

GetNextDlgTabItem

Syntax
```
HWND GetNextDlgTabItem(HWND Dlg, HWND Ctrl,
BOOL Previous);
```

Beschreibung Diese Funktion liefert das Handle des ersten Steuerelements, das den Stil *WS_TABSTOP* hat und dem vom Parameter *Ctrl* bezeichneten Steuerelement vorangeht oder darauf folgt.

Dlg: Bezeichnet das zu durchsuchende Dialogfenster.

Ctrl: Bezeichnet das Steuerelement, das als Startpunkt der Suche benutzt werden soll.

Previous: Bestimmt, wie die Funktion das Dialogfenster absuchen soll. Ist der Parameter *Previous* Null, sucht die Funktion das vorhergehende Steuerelement im Dialogfenster. Ist er ungleich Null, sucht die Funktion nach dem nächsten Steuerelement im Dialogfenster.

Rückgabewert Der Rückgabewert bezeichnet das vorherige oder nächste Steuerelement, das den Stil *WS_TABSTOP* gesetzt hat.

GetNextWindow

Syntax
```
HWND GetNextWindow(HWND Wnd, WORD Flag);
```

Beschreibung Diese Funktion sucht nach einem Handle, welches das nächste oder vorherige Fenster in der Liste des Fenster-Managers

bezeichnet. Die Liste des Fenster-Managers enthält Einträge für alle Fenster höchster Ebene, die ihnen zugehörigen untergeordneten Fenster und die untergeordneten Fenster aller untergeordneten Fenster.

Wenn der Parameter *Wnd* ein Handle zu einem Hauptfenster ist, sucht die Funktion nach dem nächsten oder vorherigen Handle zu einem Hauptfenster. Ist *Wnd* ein Handle zu einem untergeordneten Fenster, sucht die Funktion nach einem Handle zum nächsten oder vorherigen untergeordneten Fenster.

Wnd: Bezeichnet das aktuelle Fenster.

Flag: Bestimmt, ob die Funktion ein Handle zum nächsten oder zum vorherigen Fenster zurückgibt. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
GW_HWNDNEXT	Die Funktion gibt ein Handle zum nächsten Fenster zurück.
GW_HWNDPREV	Die Funktion gibt ein Handle zum vorherigen Fenster zurück.

Rückgabewert Der Rückgabewert bezeichnet das nächste oder vorherige Fenster in der Liste des Fenster-Managers.

GetNumTasks

Syntax
```
WORD GetNumTasks(void);
```

Beschreibung Diese Funktion liefert die Anzahl von Tasks, die gerade im System ablaufen. Eine Task ist eine einzelne Instanz einer Windows-Anwendung. Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert ist ein Integerwert, der die Anzahl der Tasks bezeichnet, die gerade im System ausgeführt werden.

GetObject

Syntax
```
int GetObject(HANDLE hObject, int Count, LPSTR Object);
```

Beschreibung Diese Funktion füllt einen Puffer mit den virtuellen Daten, die das vom Parameter *hObject* bezeichnete virtuelle Objekt definieren.

GetObject kopiert die vom Parameter *Count* vorgegebene Anzahl von Datenbytes in den Puffer, auf den der Parameter *Object* zeigt.

Die Funktion liefert, abhängig vom virtuellen Objekt, Datenstrukturen des Typs *LOGPEN*, *LOGBRUSH*, *LOGFONT*, *BIT-*

MAP oder einen Integerwert. Der Puffer muß ausreichend groß sein, um die Daten aufnehmen zu können.

Wenn *hObject* ein Bitmap bezeichnet, zeigt die Funktion nur die Formatinformationen über Breite, Höhe und Farbe des Bitmaps an. Die wirklichen Bits müssen mit der Funktion *GetBitmapBits* abgerufen werden.

Wenn *hObject* eine virtuelle Palette bezeichnet, liefert sie einen Zwei-Byte-Wert, der die Anzahl der Einträge in der Palette enthält. Sie liefert nicht die gesamte Datenstruktur *LOGPALETTE*, die die Palette definiert. Um Informationen über Paletteneinträge abzurufen, muß eine Anwendung die Funktion *GetPaletteEntries* aufrufen.

hObject: Bezeichnet einen Stift, Pinsel, eine Schrift oder Palette oder ein Bitmap (alle virtuell).

Count: Gibt die Anzahl von Bytes an, die in den Puffer kopiert werden sollen.

Object: Zeigt auf den Puffer, der die Informationen aufnehmen soll.

Rückgabewert Der Rückgabewert gibt die tatsächliche Anzahl abgerufener Bytes an. Er ist 0, wenn ein Fehler auftritt.

Querverweis *GetBitmapBits, GetPaletteEntries*

GetPaletteEntries

Syntax
```
WORD GetPaletteEntries(HPALETTE Palette,
WORD StartIndex, WORD NumEntries,
LPPALETTEENTRY PaletteEntries);
```

Beschreibung Diese Funktion liefert einen Bereich von Paletteneinträgen in einer virtuellen Palette.

Palette: Bezeichnet die virtuelle Palette.

StartIndex: Bezeichnet den ersten Eintrag in der virtuellen Palette, der abgerufen werden soll.

NumEntries: Gibt die Zahl der Eintrage in der virtuellen Palette an, die abgerufen werden sollen.

PaletteEntries: Zeigt auf ein Array von *PALETTEENTRY*-Datenstrukturen, das die Paletteneinträge aufnehmen sollen. Das Array muß mindestens soviele Datenstrukturen enthalten, wie vom Parameter *NumEntries* vorgegeben wird.

Rückgabewert Der Rückgabewert ist die Anzahl von Einträgen, die aus der virtuellen Palette abgerufen wurden. Tritt ein Fehler auf, ist der Rückgabewert Null.

GetParent

Syntax
```
HWND GetParent(HWND Wnd);
```

Beschreibung Diese Funktion liefert das Fenster-Handle des übergeordneten Fensters des angegebenen Fensters.

Wnd: Bezeichnet das Fenster, für welches das Handle seines übergeordneten Fensters abgerufen werden soll.

Rückgabewert Der Rückgabewert bezeichnet das übergeordnete Fenster. Wenn das Fenster kein übergeordnetes Fenster hat, wird 0 zurückgegeben.

GetPixel

Syntax
```
DWORD GetPixel(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion liefert den RGB-Farbwert des Pixels an dem durch die Parameter *X* und *Y* bezeichneten Punkt. Der Punkt muß in der Clipping-Region liegen, ist dies nicht der Fall, wird die Funktion ignoriert.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate des zu überprüfenden Punktes.

Y: Bestimmt die virtuelle y-Koordinate des zu überprüfenden Punktes.

Rückgabewert Der Rückgabewert bezeichnet einen RGB-Farbwert für die Farbe des angegebenen Punktes. Er ist -1, wenn die Koordinaten keinen Punkt in der Clipping-Region bezeichnen.

Hinweis Nicht alle Geräte unterstützen die Funktion *GetPixel*. Vergleiche hierzu die Beschreibung der Funktion *GetDeviceCaps*.

GetPolyFillMode

Syntax
```
int GetPolyFillMode(HDC DC);
```

Beschreibung Diese Funktion liefert den aktuellen Polygonfüllmodus.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert gibt den Polygonfüllmodus an. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
ALTERNATE	Füllt alternierende Flächen.
WINDING	Füllt alle Flächen.

Weitere Informationen finden sich in der Beschreibung der Funktion *SetPolyFillMode*.

Querverweis *CreatePolyPolygonRgn*

GetPriorityClipboardFormat

Syntax
```
int GetPriorityClipboardFormat(WORD FAR * PriorityList,
int Count);
```

Beschreibung Diese Funktion gibt das erste Zwischenablageformat einer Liste zurück, für das Daten existieren.

PriorityList: Zeigt auf ein Integer-Array, das eine Liste von Zwischenablageformaten enthält, die nach ihrer Priorität geordnet sind. Es handelt sich dabei um die Konstanten, die in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »CF_XXXX« beschrieben werden.

Count: Bestimmt die Anzahl der Einträge in *PriorityList*. Dieser Wert darf nicht größer als die tatsächliche Zahl von Einträgen in der Liste sein.

Rückgabewert Der Rückgabewert ist das Zwischenablageformat in der Liste mit der höchsten Priorität, für das Daten vorhanden sind. Sind in der Zwischenablage keine Daten vorhanden, gibt die Funktion 0 zurück, sind Daten vorhanden, die zu keinem Format in der Liste passen, ist der Rückgabewert -1.

GetPrivateProfileInt

Syntax
```
WORD GetPrivateProfileInt(LPSTR ApplicationName,
LPSTR KeyName, int Default, LPSTR FileName);
```

Beschreibung Diese Funktion liefert den Wert einer Integerschlüsselanweisung aus der bezeichneten Initialisierungsdatei. Sie sucht die Datei nach einer durch den Parameter *KeyName* angegebenen Schlüsselanweisung ab und überprüft dann ihre Übereinstimmung mit der Schlüsselanweisung unter dem Programmkopf der durch *ApplicationName* bezeichneten Anwendung.

Ein Integereintrag in der Initialisierungsdatei muß folgende Form haben:

```
[Anwendungsname]
Schlüsselname = Wert
.
.
.
```

ApplicationName: Zeigt auf den Namen einer Windows-Anwendung, die in der Initialisierungsdatei erscheint.

KeyName: Zeigt auf den Schlüsselnamen, der in der Initialisierungsdatei erscheint.

Default: Bestimmt den Standardwert der angegebenen Schlüsselanweisung, wenn die Schlüsselanweisung in der Initialisierungsdatei nicht gefunden werden kann.

FileName: Zeigt auf einen nullterminierten String, der die Initialisierungsdatei bezeichnet. Enthält *FileName* keine Pfadangabe für die Datei, sucht Windows im Windows-Verzeichnis nach der Datei.

Rückgabewert Der Rückgabewert ist 0, wenn der zum angegebenen Schlüsselnamen gehörende Wert kein Integerwert oder ein negativer Integerwert ist. Besteht der Wert, der zum Schlüsselnamen gehört, aus Ziffern, auf die nichtnumerische Zeichen folgen, gibt die Funktion den Wert der Ziffern zurück. Wird zum Beispiel auf den Eintrag *KeyName=102abc* zugegriffen, gibt die Funktion *102* zurück. Wird die Schlüsselanweisung nicht gefunden, ist der Rückgabewert der Standardwert *Default*.

Hinweis Die Funktion *GetPrivateProfileInt* berücksichtigt Groß- und Kleinschreibung nicht. Aus diesem Grund können die Strings in *ApplicationName* und *KeyName* aus einer beliebigen Kombination von großen und kleinen Buchstaben bestehen.

GetPrivateProfileString

Syntax
```
int GetPrivateProfileString(LPSTR ApplicationName,
LPSTR KeyName, LPSTR Default, LPSTR ReturnedString,
int Size, LPSTR FileName);
```

Beschreibung Diese Funktion kopiert einen String aus der angegebenen Initialisierungsdatei in einen Puffer, auf den der Parameter *ReturnedString* zeigt. Sie durchsucht die Datei nach einer Schlüsselanweisung, die mit der vom Parameter *KeyName* vorgegebenen Bezeichnung unter dem Programmkopf der vom Parameter *ApplicationName* vorgegebenen Anwendung übereinstimmt. Wird diese gefunden, wird der zugehörende String in den Puffer kopiert. Ist die Schlüsselanweisung nicht vorhanden, wird die durch den Parameter *Default* bezeichnete Standardzeichenkette in den Puffer kopiert.

Ein Stringeintrag in der Initialisierungsdatei muß folgende Form haben:

```
[Anwendungsname]
Schlüsselname = string
.
.
.
```

Ist *KeyName* NULL, listet *GetPrivateProfileString* alle Schlüsselnamen auf, die *ApplicationName* zugeordnet sind, indem sie die Stelle, auf die *ReturnedString* zeigt, mit einer Liste von Schlüsselnamen füllt (nicht mit Werten). Jede Schlüsselanweisung in der Liste ist nullterminiert.

ApplicationName: Zeigt auf den Namen einer Windows-Anwendung, der in der Initialisierungsdatei erscheint.

KeyName: Zeigt auf einen Schlüsselnamen, der in der Initialisierungsdatei erscheint.

Default: Zeigt auf den Standardwert der bezeichneten Schlüsselanweisung, wenn die Schlüsselanweisung in der Initialisierungsdatei nicht gefunden werden kann.

ReturnedString: Zeigt auf den Puffer, der den String aufnehmen soll.

Size: Gibt die maximale Anzahl von Zeichen an (einschließlich Nullzeichen), die in den Puffer kopiert werden sollen.

FileName: Zeigt auf einen String, der die Initialisierungsdatei benennt. Enthält *FileName* keine Pfadangabe für die Datei, sucht Windows die Datei im Windows-Verzeichnis.

Rückgabewert Der Rückgabewert gibt die Anzahl der in den vom Parameter *ReturnedString* bezeichneten Puffer kopierten Zeichen an. Die abschließenden Nullzeichen werden dabei nicht berücksichtigt. Wenn *KeyName* nicht NULL und der Puffer nicht groß genug ist, um den ganzen String aufzunehmen, ist der Rückgabewert gleich der vom Parameter *Size* vorgegebenen Länge. Ist *KeyName* NULL und der Puffer nicht groß genug, um den ganzen String aufzunehmen, ist der Rückgabewert um 2 kleiner als die vom Parameter *Size* vorgegebene Länge.

Hinweis *GetPrivateProfileString* berücksichtigt Groß- und Kleinschreibung nicht, so daß die Strings in *ApplicationName* und *KeyName* aus einer beliebigen Kombination von großen und kleinen Buchstaben bestehen können.

GetProcAddress

Syntax
```
FARPROC GetProcAddress(HANDLE Module, LPSTR ProcName);
```

Beschreibung Diese Funktion liefert die Speicheradresse der Funktion, auf deren Bezeichnung der Parameter *ProcName* zeigt.

Module: Bezeichnet das Bibliotheksmodul, welches die Funktion enthält.

ProcName: Zeigt auf die Funktionsbezeichnung oder enthält die Ordnungszahl der Funktion. Wenn es eine Ordnungszahl

ist, so muß diese im niederwertigen und 0 im höherwertigen WORD liegen. *ProcName* muß auf einen nullterminierten String zeigen.

Rückgabewert Bei erfolgreicher Ausführung zeigt der Rückgabewert auf den Einsprungspunkt der Funktion, andernfalls ist er 0.

Ist der Parameter *ProcName* eine Ordnungszahl und existiert keine Funktion mit der angegebenen Ordnungszahl in dem Modul, kann *GetProcAddress* trotzdem noch einen von 0 verschiedenen Wert zurückgeben. Man sollte daher in den Fällen, in denen die Funktion eventuell nicht existiert, die Funktion mit der Bezeichnung anstatt mit der Ordnungszahl benennen.

Hinweis *GetProcAddress* sollte nur dazu verwendet werden, Adressen von exportierten Funktionen abzurufen, die zu Bibliotheksmodulen gehören. Die Funktion *MakeProcInstance* kann dazu verwendet werden, auf Funktionen innerhalb unterschiedlicher Instanzen des aktuellen Moduls zuzugreifen.

GetProfileInt

Syntax
```
WORD GetProfileInt(LPSTR AppName, LPSTR KeyName,
int Default);
```

Beschreibung Diese Funktion liefert den Wert einer Integerschlüsselanweisung aus der Windows-Initialisierungsdatei WIN.INI.

GetProfileInt durchsucht die Datei WIN.INI nach einer Schlüsselanweisung unter dem durch den Parameter *AppName* bezeichneten Anwendungsprogrammkopf, die mit der durch den Parameter *KeyName* benannten Bezeichnung übereinstimmt.

Ein Integereintrag in WIN.INI muß folgende Form haben:

```
[Anwendungsname]
Schlüsselname = Wert
.
.
.
```

AppName: Zeigt auf die Bezeichnung einer Windows-Anwendung, die in der Windows-Initialisierungsdatei erscheint.

KeyName: Zeigt auf einen Schlüsselnamen, der in der Windows-Initialisierungsdatei erscheint.

Default: Bestimmt den Standardwert für die gegebene Schlüsselanweisung, wenn die Schlüsselanweisung in der Windows-Initialisierungsdatei nicht gefunden werden kann.

Rückgabewert Der Rückgabewert ist 0, wenn der zur angegebenen Schlüsselbezeichnung gehörende Wert kein Integerwert oder ein negativer Integerwert ist. Besteht der Wert, der zum Schlüsselna-

men gehört, aus Ziffern, auf die nichtnumerische Zeichen folgen, gibt die Funktion den Wert der Ziffern zurück.

Wenn z.B. auf den Eintrag *KeyName=102abc* zugegriffen wird, gibt die Funktion *102* zurück. Wird die Schlüsselanweisung nicht gefunden, gibt sie den Standardwert *Default* zurück.

GetProfileString

Syntax

```
int GetProfileString(LPSTR AppName, LPSTR KeyName,
LPSTR Default, LPSTR ReturnedString, int Size);
```

Beschreibung

Diese Funktion kopiert einen String aus der Windows-Initialisierungsdatei WIN.INI in den Puffer, auf den der Parameter *ReturnedString* zeigt.

GetProfileString durchsucht WIN.INI nach einer Schlüsselanweisung unter dem durch den Parameter *AppName* bezeichneten Anwendungsprogrammkopf, die mit der durch den Parameter *KeyName* benannten Bezeichnung übereinstimmt.

Wenn die Schlüsselanweisung gefunden wird, dann wird der entsprechende String in den Puffer kopiert, andernfalls wird die im Parameter *Default* angegebene Voreinstellung kopiert.

Ein Stringeintrag in WIN.INI muß die folgende Form haben:

```
[Anwendungsname]
Schlüsselname = Wert
.
.
.
```

Ist *KeyName* NULL, listet die Funktion *GetProfileString* alle Schlüsselnamen auf, die *AppName* zugeordnet sind, indem sie die Position, auf die der Parameter *ReturnedString* zeigt, mit einer Liste von Schlüsselnamen (nicht Werten) füllt. Jeder Schlüsselname in der Liste ist mit einem Nullzeichen abgeschlossen.

AppName: Zeigt auf einen nullterminierten String, der die Anwendung bezeichnet.

KeyName: Zeigt auf einen nullterminierten String, der eine Schlüsselanweisung bezeichnet.

Default: Zeigt auf den voreingestellten Wert für die angegebene Schlüsselanweisung, wenn die Schlüsselanweisung in der Initialisierungsdatei nicht gefunden werden kann.

ReturnedString: Zeigt auf den Puffer, der die String empfängt.

Size: Bestimmt die Anzahl der Zeichen (einschließlich des letzten Nullzeichens), die in den Puffer kopiert werden.

Rückgabewert Der Rückgabewert gibt die Anzahl der in den vom Parameter *ReturnedString* bezeichneten Puffer kopierten Zeichen an, ohne Berücksichtigung des abschließenden Nullzeichens. Wenn der Puffer nicht groß genug ist, um den gesamten String aufzunehmen, und *KeyName* ungleich Null ist, ist der Rückgabewert gleich der vom Parameter *Size* angegebenen Länge. Wenn der Puffer nicht groß genug ist, um den gesamten String aufzunehmen, und *KeyName* NULL ist, ist der Rückgabewert um 2 kleiner als die vom Parameter *Size* vorgegebene Länge.

Hinweis *GetProfileString* läßt Groß- und Kleinschreibung unberücksichtigt, so daß die Strings in *AppName* und *KeyName* in einer beliebigen Kombination von Groß- und Kleinschreibung eingegeben werden können.

GetProp

Syntax
```
HANDLE GetProp(HWND Wnd, LPSTR Str);
```

Beschreibung Diese Funktion liefert ein Daten-Handle aus der Eigenschaftsliste des angegebenen Fensters. Der String, auf den der Parameter *Str* zeigt, bezeichnet das Handle, das geliefert werden soll.

Es wird vorausgesetzt, daß der String und das Handle durch die Funktion *SetProp* der Eigenschaftsliste hinzugefügt wurden.

Wnd: Bezeichnet das Fenster, dessen Eigenschaftsliste durchsucht werden soll.

Str: Zeigt auf einen nullterminierten String oder ein Atom, das einen String bezeichnet. Wenn ein Atom angegeben ist, muß dies vorher mit der Funktion *AddAtom* erzeugt worden sein. Das Atom, ein 16-Bit-Wert, muß im niederwertigen WORD des Parameters *Str* positioniert werden, das höherwertige WORD muß auf 0 gesetzt werden.

Rückgabewert Wenn die Eigenschaftsliste den angegebenen String enthält, bezeichnet der Rückgabewert das zugehörende Daten-Handle, andernfalls ist er 0.

Hinweis Der von der Funktion *GetProp* abgerufene String kann jeder für die Anwendung verwendbare 16-Bit-Wert sein.

Querverweis *AddAtom*

GetRgnBox

Syntax
```
int GetRgnBox(HRGN Rgn, LPRECT Rect);
```

Beschreibung Diese Funktion liefert die Koordinaten des umgrenzenden Rechtecks der durch den Parameter *Rgn* bezeichneten Region.

Rgn: Bezeichnet die Region.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die Koordinaten des umgrenzenden Rechtecks aufnehmen soll.

Rückgabewert Der Rückgabewert gibt den Typ der Region an. Er kann einen der folgenden Werte annehmen.

Wert	Bedeutung
COMPLEXREGION	Region hat überlappende Grenzen.
NULLREGION	Region ist leer.
SIMPLEREGION	Region hat keine überlappenden Grenzen.

Der Rückgabewert ist 0, wenn der Parameter *Rgn* keine gültige Region bezeichnet.

GetROP2

Syntax
```
int GetROP2(HDC DC);
```

Beschreibung Diese Funktion liefert den aktuellen Zeichenmodus.

Der Zeichenmodus legt fest, wie der Stift oder die innere Farbe und die bereits auf der Bildschirmfläche befindliche Farbe kombiniert werden sollen, um eine neue Farbe zu ergeben.

DC: Bezeichnet den Gerätekontext für ein rasterorientiertes Gerät.

Rückgabewert Der Rückgabewert gibt den Zeichenmodus an. Der Abschnitt »R2_XXXX« in Kapitel 4 »Windows-Stile und -Konstanten« enthält die möglichen Werte. Zusätzliche Informationen finden sich auch in der Beschreibung der Funktion *SetROP2*.

Querverweis *SetROP2*

GetRValue

Syntax
```
BYTE GetRValue(DWORD RGBColor);
```

Beschreibung Dieses Makro entnimmt einem RGB-Farbwert den Wert für Rot.

RGBColor: Bezeichnet ein rotes, ein blaues und ein grünes Farbfeld. Jedes der Felder bestimmt die Intensität der angegebenen Farbe.

Rückgabewert Der Rückgabewert bezeichnet ein Byte, das den roten RGB-Farbwert enthält.

Hinweis Der Wert 0x0FF entspricht der maximalen, 0x000 der minimalen Intensität für ein einzelnes Byte.

GetScrollPos

Syntax
```
int GetScrollPos(HWND Wnd, int Bar);
```

Beschreibung Diese Funktion liefert die aktuelle Position eines Positionszeigers. Die aktuelle Position ist ein relativer Wert, der vom aktuellen Bildlaufbereich abhängt.

Geht der Bildlaufbereich zum Beispiel von 0 bis 100 und ist das Symbol in der Mitte der Leiste, ist die aktuelle Position 50.

Wnd: Bezeichnet – abhängig vom Wert des Parameters *Bar* – ein Fenster mit Standardbildlaufleisten oder eine selbständige Bildlaufleiste.

Bar: Bestimmt die Bildlaufleiste, die überprüft werden soll. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
SB_CTRL	Liefert die Position einer selbständigen Bildlaufleiste. In diesem Fall muß der Parameter *Wnd* das Fenster-Handle einer Bildlaufleiste sein.
SB_HORZ	Liefert die Position der horizontalen Bildlaufleiste eines Fensters.
SB_VERT	Liefert die Position der vertikalen Bildlaufleiste eines Fensters.

Rückgabewert Gibt die aktuelle Position des Positionszeigers an.

GetScrollRange

Syntax
```
void GetScrollRange(HWND Wnd, int Bar, LPINT MinPos,
LPINT MaxPos);
```

Beschreibung Diese Funktion kopiert die für die angegebene Bildlaufleiste aktuellen maximalen und minimalen Positionen des Positionszeigers in die Parameter *MinPos* und *MaxPos*.

Wenn das angegebene Fenster keine Stanardbildlaufleisten hat oder keine selbständige Bildlaufleiste ist, kopiert *GetScrollRange* 0 in *MinPos* und *MaxPos*.

Wnd: Bezeichnet ein Fenster mit Standardbildlaufleisten oder eine selbständige Bildlaufleiste, abhängig vom Wert des Parameters *Bar*.

Bar: Bestimmt einen Integerwert, der die abzurufende Bildlaufleiste bezeichnet.

Möglich ist einer der in der folgendenden Tabelle dargestellten Werte:

Wert	Bedeutung
SB_CTRL	Liefert die Position einer selbständigen Bildlaufleiste, in diesem Fall muß der Parameter *Wnd* das Handle einer Bildlaufleiste sein.
SB_HORZ	Liefert die Position der horizontalen Bildlaufleiste eines Fensters.
SB_VERT	Liefert die Position der vertikalen Bildlaufleiste eines Fensters.

MinPos: Integervariable, die die minimale Position aufnehmen soll.

MaxPos: Integervariable, die die maximale Position aufnehmen soll.

Rückgabewert Keiner.

Hinweis Der voreingestellte Bereich für eine Standardbildlaufleiste geht von 0 bis 100. Der voreingestellte Bereich für eine selbständige Bildlaufleiste ist leer (beide Werte sind 0).

GetStockObject

Syntax
```
HANDLE GetStockObject(int Index);
```

Beschreibung Diese Funktion liefert ein Handle zu einem der vordefinierten Stifte, Pinsel oder Schriften.

Index: Bestimmt den Typ des gewünschten vordefinierten Objekts. Dieser Parameter kann einen der folgenden Werte annehmen:

Wert	Bedeutung
BLACK_BRUSH	Schwarzer Pinsel.
DKGRAY_BRUSH	Dunkelgrauer Pinsel.
GRAY_BRUSH	Grauer Pinsel.
HOLLOW_BRUSH	Hohler Pinsel.
LTGRAY_BRUSH	Hellgrauer Pinsel.
NULL_BRUSH	Neutraler Pinsel.
WHITE_BRUSH	Weißer Pinsel.
BLACK_PEN	Schwarzer Stift.
NULL_PEN	Null-Stift.
WHITE_PEN	Weißer Stift.
ANSI_FIXED_FONT	ANSI-Systemschrift mit festen Zeichenabständen.
ANSI_VAR_FONT	ANSI-Systemschrift mit variablen Zeichenabständen.
DEVICE_DEFAULT_FONT	Geräteabhängige Schrift.

Wert	Bedeutung
OEM_FIXED_FONT	OEM-Schrift mit festen Zeichenabständen.
SYSTEM_FONT	Die Systemschrift, die von Windows standardmäßig benutzt wird, um Menüs, Dialogfenster und anderen Text darzustellen. In den Windows-Versionen 3.0 und höher ist die Systemschrift eine Proportionalschrift. Frühere Versionen benutzen eine Systemschrift mit festem Zeichenabstand.
SYSTEM_FIXED_FONT	Die Systemschrift mit festem Zeichenabstand die in früheren Windows-Versionen benutzt wurde. Dieses Objekt ist nur aus Gründen der Kompatibilität verfügbar.
DEFAULT_PALETTE	Standardfarbpalette. Sie besteht aus den 20 statischen Farben, die immer in der Systempalette präsent sind, um Farben in der virtuellen Palette von Hintergrundfenstern anzugleichen.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt werden konnte, bezeichnet der Rückgabewert das gewünschte virtuelle Objekt, andernfalls ist er 0.

Hinweis Die Objekte *DKGRAY_BRUSH*, *GRAY_BRUSH* und *LTGRAY_BRUSH* sollten nicht als Hintergrundpinsel oder für einen anderen Zweck in einem Fenster verwendet werden, dessen Klasse nicht die Stile *CS_HREDRAW* und *CS_VREDRAW* enthält. Die Verwendung eines grauen vordefinierten Pinsels in solchen Fenstern kann zu Fehlausrichtungen von Pinselmustern führen, wenn ein Fenster verschoben oder in der Größe neu festgelegt wurde. Die Ursprünge vordefinierter Pinsel können nicht eingestellt werden (vgl. die Beschreibung der Funktion *SetBrushOrg*).

GetStretchBltMode

Syntax
```
int GetStretchBltMode(HDC DC);
```

Beschreibung Diese Funktion liefert den aktuellen Streckmodus. Dieser definiert, wie Informationen zu Bitmaps hinzugefügt oder daraus gelöscht werden, die mit der Funktion *StretchBlt* erweitert oder komprimiert werden.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert bestimmt den aktuellen Streckmodus. Er kann *WHITEONBLACK*, *BLACKONWHITE* oder *COLORONCOLOR* sein (siehe Abschnitt »Streckmodi« in Kapitel 4 »Windows-Stile und -Konstanten«.

Querverweis *SetStretchBltMode*

GetSubMenu

Syntax
```
HMENU GetSubMenu(HMENU Menu, int Pos);
```

Beschreibung Diese Funktion liefert das Menü-Handle eines Pop-up-Menüs.

Menu: Bezeichnet das Menü.

Pos: Bezeichnet die Position des Pop-up-Menüs im angegebenen Menü. Positionswerte beginnen bei 0 für die erste Menüoption. Die Integer-ID des Pop-up-Menüs kann in dieser Funktion nicht verwendet werden.

Rückgabewert Der Rückgabewert bezeichnet das Pop-up-Menü. Er ist 0, wenn an der angegebenen Position kein Pop-up-Menü existiert.

GetSysColor

Syntax
```
DWORD GetSysColor(int Index);
```

Beschreibung Diese Funktion liefert die aktuelle Farbe des durch *Index* bezeichneten Bildschirmelements. Bildschirmelemente sind die verschiedenen Teile eines Fensters und des Windows-Bildschirms, die auf dem Systembildschirm erscheinen.

Index: Bestimmt das Bildschirmelement, dessen Farbe abgerufen werden soll. Eine Liste der Indexwerte findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »COLOR_XXXX« und bei der Beschreibung der Funktion *SetSysColors*.

Rückgabewert Der Rückgabewert gibt einen RGB-Farbwert an, der die Farbe des angegebenen Elements bezeichnet.

Hinweis Systemfarben für Monochrombildschirme werden normalerweise als unterschiedliche Grauschattierungen interpretiert.

Querverweis *SetSysColor*

GetSysModalWindow

Syntax
```
HWND GetSysModalWindow(void);
```

Beschreibung Diese Funktion gibt das Handle eines systemmodalen Fensters zurück, wenn ein solches vorhanden ist.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert bezeichnet das systemmodale Fenster, falls ein solches vorhanden ist, wenn nicht, dann ist der Rückgabewert 0.

GetSystemDirectory

Syntax
```
WORD GetSystemDirectory(LPSTR Buffer, WORD Size);
```

Beschreibung Diese Funktion ermittelt den Pfadnamen des Windows-Systemunterverzeichnisses. Das Systemunterverzeichnis enthält Dateien wie Bibliotheken, Treiber und Schriftdateien.

Buffer: Zeigt auf den Puffer, der den nullterminierten String aufnehmen soll, die den Pfadnamen enthält.

Size: Bestimmt die maximale Größe des Puffers in Bytes. Dieser Wert sollte mindestens auf 144 gesetzt werden, um im Puffer genügend Platz für den Pfadnamen zu schaffen.

Rückgabewert Der Rückgabewert ist die Länge des in den Parameter *Buffer* kopierten Strings, wobei das abschließende Nullzeichen nicht mitgezählt wird.

Ist der Rückgabewert größer als *Size*, ist er die Größe des Puffers, die erforderlich ist, um den Pfadnamen aufzunehmen. Der Rückgabewert ist 0, wenn die Funktion fehlschlägt.

Hinweis Der von dieser Funktion abgerufene Pfadname endet nicht mit einem Backslash (\), außer wenn das Systemverzeichnis das Hauptverzeichnis des Datenträgers ist.

Wenn das Systemverzeichnis zum Beispiel den Namen *WINDOWS\SYSTEM* in Laufwerk C: hat, lautet der von der Funktion abgerufene Pfadname des Systemunterverzeichnisses *C:\WINDOWS\SYSTEM*.

GetSystemMenu

Syntax
```
HMENU GetSystemMenu(HWND Wnd, BOOL Revert);
```

Beschreibung Diese Funktion erlaubt es der Anwendung, auf das Systemmenü zuzugreifen, um dieses zu kopieren und/oder zu verändern.

Wnd: Bezeichnet das Fenster, welches eine Kopie des Systemmenüs aufnehmen soll.

Revert: Gibt die auszuführende Aktion an. Ist *Revert* Null, gibt *GetSystemMenu* ein Handle einer Kopie des aktuell verwendeten Systemmenüs zurück. Diese Kopie ist anfangs mit dem Systemmenü identisch, kann aber verändert werden.

Ist *Revert* ungleich Null, entfernt *GetSystemMenu* die mögli-

cherweise veränderte Kopie des Systemmenüs, die zu dem angegebenen Fenster gehört und gibt ein Handle zu der ursprünglichen, unveränderten Version des Systemmenüs zurück.

Rückgabewert Wenn *Revert* ungleich 0 ist und das Systemmenü verändert wurde, bezeichnet der Rückgabewert das Systemmenü.

Wenn *Revert* ungleich Null ist und das Systemmenü nicht verändert wurde, ist der Rückgabewert 0. Ist *Revert* Null, bezeichnet der Rückgabewert eine Kopie des Systemmenüs.

Hinweis Jedes Fenster, das nicht *GetSystemMenu* verwendet, um eine eigene Kopie des Systemmenüs zu erstellen, erhält ein Standardsystemmenü.

Das von *GetSystemMenu* zurückgegebene Handle kann mit den Funktionen *AppendMenu*, *InsertMenu* oder *ModifyMenu* verwendet werden, um das Systemmenü zu ändern. Das Systemmenü enthält anfänglich Optionen, die durch verschiedene ID-Werte – *SC_CLOSE*, *SC_MOVE* und *SC_SIZE* – bezeichnet werden.

Menüoptionen im Systemmenü senden *WM_SYSCOMMAND*-Botschaften. Alle vordefinierten Systemmenüoptionen haben ID-Nummern, die größer als 0xF000 sind. Fügt eine Anwendung Befehle zum Systemmenü hinzu, sollte sie ID-Nummern kleiner als 0xF000 vergeben.

Windows versetzt, abhängig von der Situation, Optionen im Standardsystemmenü automatisch in den nicht-aktivierbaren Zustand. Die Anwendung kann durch Reaktion auf die Botschaft *WM_INITMENU* ihre eigene Auswahl oder ihre eigene Versetzung in den nicht aktivierbaren Zustand herstellen. Diese Botschaft wird vor der Darstellung eines Menüs übergeben.

Querverweis *AppendMenu, InsertMenu, ModifyMenu*

GetSystemMetrics

Syntax `int GetSystemMetrics(int Index);`

Beschreibung Diese Funktion liefert die Systemmaße. Das sind die Breiten und Höhen verschiedener Bildschirmelemente des Windows-Bildschirms. *GetSystemMetrics* kann auch Flags zurückgeben, die anzeigen, ob die aktuelle Version eine Debugger-Version ist, ob eine Maus angeschlossen ist, oder ob die Bedeutung der linken und rechten Maustaste ausgetauscht wurde.

Index: Bestimmt die Systemmaßeinheit, die abgerufen werden soll. Alle Maßeinheiten werden in Pixeln angegeben.

Die Systemmaßeinheit ist einer der im folgenden »Hinweis« beschriebenen Werte.

Rückgabewert Der Rückgabewert gibt die angeforderten Systemmaßeinheiten an.

Hinweis Systemmaße hängen vom Systembildschirm ab und können von Bildschirm zu Bildschirm schwanken. Die Werte für den Parameter *Index* sind in der folgenden Übersicht kurz beschrieben:

Index	Bedeutung
SM_CXSCREEN	Breite des Bildschirms.
SM_CYSCREEN	Höhe des Bildschirms.
SM_CXFRAME	Breite des einstellbaren Fensterrahmens.
SM_CYFRAME	Höhe des einstellbaren Fensterrahmens.
SM_CXVSCROLL	Breite des Pfeilbitmaps auf der vertikalen Bildlaufleiste.
SM_CYVSCROLL	Höhe des Pfeilbitmaps auf der vertikalen Bildlaufleiste.
SM_CXHSCROLL	Breite des Pfeilbitmaps auf der horizontalen Bildlaufleiste.
SM_CYHSCROLL	Höhe des Pfeilbitmaps auf der horizontalen Bildlaufleiste.
SM_CYCAPTION	Höhe der Titelzeile eines Fensters.
SM_CXBORDER	Breite des nicht einstellbaren Fensterrahmens.
SM_CYBORDER	Höhe des nicht einstellbaren Fensterrahmens.
SM_CXDLGFRAME	Breite des Rahmens, wenn das Fenster den Stil *WS_DLGFRAME* hat.
SM_CYDLGFRAME	Höhe des Rahmens, wenn das Fenster den Stil *WS_DLGFRAME* hat.
SM_CXHTHUMB	Breite des Positionszeigers auf der horizontalen Bildlaufleiste.
SM_CYVTHUMB	Höhe des Positionszeigers auf der vertikalen Bildlaufleiste.
SM_CXICON	Symbolbreite.
SM_CYICON	Symbolhöhe.
SM_CXCURSOR	Breite des Mauszeigers.
SM_CYCURSOR	Höhe des Mauszeigers.
SM_CYMENU	Höhe einer einzeiligen Menüleiste.
SM_CXFULLSCREEN	Breite des Fenster-Client-Bereiches für eine Vollbildanwendung.
SM_CYFULLSCREEN	Höhe des Fenster-Client-Bereiches für eine Vollbildanwendung (gleich der Höhe des Bildschirms minus der Höhe der Titelzeile).
SM_CYKANJIWINDOW	Höhe des Kanji-Fensters.
SM_CXMINTRACK	Minimale Tracking-Breite des Fensters.
SM_CYMINTRACK	Minimale Tracking-Höhe des Fensters.
SM_CXMIN	Minimale Breite des Fensters.

Index	Bedeutung
SM_CYMIN	Minimale Höhe des Fensters.
SM_CXSIZE	Breite der Bitmaps, die in der Titelzeile enthalten sind.
SM_CYSIZE	Höhe der Bitmaps, die in der Titelzeile enthalten sind.
SM_MOUSEPRESENT	Ungleich 0, wenn eine Maus installiert ist.
SM_DEBUG	Ungleich 0, wenn die Windows-Debugger-Version vorliegt.
SM_SWAPBUTTON	Ungleich 0, wenn die linke und rechte Maustaste vertauscht wurden.

GetSystemPaletteEntries

Syntax

```
WORD GetSystemPaletteEntries(HDC DC, WORD StartIndex,
WORD NumEntries, LPPALETTEENTRY PaletteEntries);
```

Beschreibung Diese Funktion liefert einen Bereich von Paletteneinträgen aus der Systempalette.

DC: Bezeichnet den Gerätekontext.

StartIndex: Bestimmt den ersten Eintrag in der Systempalette, der abgerufen werden soll.

NumEntries: Bestimmt die Anzahl von Einträgen in der Systempalette, die abgerufen werden sollen.

PaletteEntries: Zeigt auf ein Array von *PALETTEENTRY*-Datenstrukturen, das die Paletteneinträge aufnehmen soll. Das Array muß mindestens soviele Datenstrukturen, wie im Parameter *NumEntries* angegeben ist, enthalten.

Rückgabewert Der Rückgabewert ist die Anzahl von Einträgen, die aus der Systempalette abgerufen wurden. Er ist 0, wenn die Funktion fehlschlägt.

GetSystemPaletteUse

Syntax

```
WORD GetSystemPaletteUse(HDC DC);
```

Beschreibung Diese Funktion ermittelt, ob eine Anwendung auf die ganze Systempalette zugreifen kann. Standardmäßig enthält die Systempalette 20 statische Farben, die nicht verändert werden, wenn eine Anwendung ihre virtuelle Palette realisiert. Eine Anwendung kann durch Aufruf der Funktion *SetSystemPaletteUse* Zugriff auf die meisten dieser Farben erlangen.

Der vom Parameter *DC* bezeichnete Gerätekontext muß sich auf ein Gerät beziehen, das Farbpaletten unterstützt.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert bestimmt die aktuelle Verwendung der Systempalette. Er hat einen der folgenden Werte:

Wert	Bedeutung
SYSPAL_NOSTATIC	Die Systempalette enthält außer schwarz und weiß keine statischen Farben.
SYSPAL_STATIC	Die Systempalette enthält statische Farben, die sich nicht ändern, wenn eine Anwendung ihre virtuelle Palette realisiert.

Querverweis *SetSystemPaletteUse*

GetTabbedTextExtent

Syntax
```
DWORD GetTabbedTextExtent(HDC DC, LPSTR Str, int Count,
int TabPositions, LPINT TabStopPositions);
```

Beschreibung Diese Funktion berechnet Breite und Höhe der Textzeile, auf die der Parameter *Str* zeigt. Enthält der String ein oder mehrere Tabulatorzeichen, richtet sich die Breite des Strings nach den Tabulatorpositionen, die vom Parameter *TabStopPositions* vorgegeben sind. Die Funktion *GetTabbedTextExtent* verwendet die aktuell ausgewählte Schrift, um die Abmessungen des Strings zu berechnen. Breite und Höhe werden ohne Berücksichtigung der aktuellen Clipping-Region in virtuellen Einheiten berechnet.

DC: Bezeichnet den Gerätekontext.

Str: Zeigt auf einen Textstring.

Count: Bestimmt die Anzahl von Zeichen im Textstring.

TabPositions: Bestimmt die Anzahl von Tabulatorstoppositionen in dem Array, auf das der Parameter *TabStopPositions* zeigt.

TabStopPositions: Zeigt auf ein Array von Integerwerten, das die Tabulatorstoppositionen in Pixeln enthält. Die Tabulatorstops müssen in aufsteigender Reihenfolge sortiert werden, Rückwärtstabulatoren sind nicht erlaubt.

Rückgabewert Der Rückgabewert gibt die Abmessungen des Strings an. Die Höhe liegt im höherwertigen, die Breite im niederwertigen WORD.

Hinweis Da einige Geräte Zeichen nicht in regulären Zellen plazieren (das heißt, sie führen *Kerning* aus), kann die Summe der Abmessungen der Zeichen in einem String eventuell von den Abmessungen des Strings verschieden sein.

Wenn die Parameter *TabPositions* und *TabStopPositions* 0 sind, werden die Tabulatoren auf acht durchschnittliche Zeichenbreiten erweitert.

Wenn *TabPositions* 1 ist, werden die Tabulatorstops mit dem Abstand getrennt, den der erste Wert in dem Array angibt, auf das *TabStopPositions* zeigt.

Wenn *TabStopPositions* auf mehr als einen einzigen Wert zeigt, wird ein Tabulatorstop für jeden Wert im Array gesetzt, bis zu der Zahl, die von *TabPositions* vorgegeben ist.

GetTempDrive

Syntax
```
BYTE GetTempDrive(BYTE DriveLetter);
```

Beschreibung Diese Funktion nimmt eine Laufwerksbezeichnung oder 0 entgegen und gibt einen Kennbuchstaben zurück, der das optimale Laufwerk für eine temporäre Datei bezeichnet. Dies ist das Diskettenlaufwerk, das während Diskettenoperationen mit einer temporären Datei die beste Zugriffszeit bietet.

GetTempDrive gibt die Laufwerksbezeichnung einer Festplatte zurück, wenn das System eine hat. Ist der Parameter *DriveLetter* 0, gibt die Funktion die Laufwerksbezeichnung der aktuellen Diskette zurück. Ist *DriveLetter* ein Buchstabe, gibt die Funktion den Kennbuchstaben dieses Laufwerks oder den Kennbuchstaben eines anderen verfügbaren Laufwerks zurück. Zur Bedeutung des Parameters *DriveLetter*, siehe »Hinweis« unter *GetTempFileName*.

DriveLetter: Bestimmt ein Laufwerk.

Rückgabewert Der Rückgabewert zeigt das optimale Laufwerk für temporäre Dateien an.

Querverweis *GetTempFileName*

GetTempFileName

Syntax
```
int GetTempFileName(BYTE DriveLetter, LPSTR
PrefixString, WORD Unique, LPSTR TempFileName);
```

Beschreibung Diese Funktion erzeugt einen temporären Dateinamen in der folgenden Form:

```
d:\Pfad\Präfixuuuu.tmp
```

In dieser Syntaxzeile ist *d:* der vom Parameter *DriveLetter* vorgegebene Kennbuchstabe. *Pfad* ist der Pfadname für die temporäre Datei (entweder das Stammverzeichnis des angegebenen Laufwerks oder das in der Environment-Variablen TEMP benannte Verzeichnis); *Präfix* sind alle Buchstaben (bis zu drei) des Strings, auf den der Parameter *PrefixString* zeigt,

uuuu ist der numerische Wert des vom Parameter *Unique* vorgegebenen Dateinamens.

DriveLetter: Bestimmt das für die temporäre Datei empfohlene Laufwerk. Wenn *DriveLetter* 0 ist, wird das Standardlaufwerk benutzt.

PrefixString: Zeigt auf einen nullterminierten String, der als Namenspräfix der temporären Datei verwendet werden soll. Dieser String muß aus Zeichen des OEM-Zeichensatzes bestehen.

Unique: Numerischer Wert des Dateinamens oder 0, wenn der Wert vom System bereitgestellt wird.

TempFileName: Zeigt auf den Puffer, welcher den temporären Dateinamen aufnehmen soll. Dieser String besteht aus Zeichen des OEM-Zeichensatzes. Der Puffer sollte mindestens 144 Bytes lang sein, um genügend Raum für den Pfadnamen zu bieten.

Rückgabewert

Der Rückgabewert gibt einen eindeutigen numerischen Wert an, der im temporären Dateinamen benutzt wird. Wenn ein Wert ungleich 0 für den Parameter *Unique* angegeben war, bestimmt der Rückgabewert denselben Wert.

Hinweis

Um Probleme zu vermeiden, die aus der Umwandlung von OEM-Zeichen in einen ANSI-String entstehen können, sollte die Anwendung die Funktion *_lopen* aufrufen, um die temporäre Datei zu erzeugen.

GetTempFileName benutzt die empfohlene Laufwerksbezeichnung zum Erzeugen des temporären Dateinamens, außer in den folgenden Fällen:

- Ist eine Festplatte vorhanden, benutzt *GetTempFileName* immer die Laufwerksbezeichnung der ersten Festplatte.
- Ist eine Environment-Variable *TEMP* definiert, deren Wert mit einer Laufwerksbezeichnung beginnt, wird dieser Kennbuchstabe verwendet.

Wenn das Bit *TF_FORCEDRIVE* des Parameters *DriveLetter* gesetzt ist, gelten die oben gemachten Ausnahmen nicht. Der temporäre Dateiname wird immer im aktuellen Verzeichnis des von *DriveLetter* vorgegebenen Laufwerks erzeugt – unabhängig davon, ob eine Festplatte oder die Environment-Variable *TEMP* vorhanden sind.

Ist der Parameter *Unique* 0, versucht *GetTempFileName*, eine auf der aktuellen Systemzeit basierende eindeutige Zahl zu bilden. Existiert bereits eine Datei mit dem sich ergebenden Dateinamen, wird die Zahl um 1 erhöht und der Test auf Vorhandensein wiederholt.

Dies wird solange durchgeführt, bis ein eindeutiger Dateiname gefunden ist, *GetTempFileName* erzeugt dann eine Datei

mit diesem Namen und schließt sie. Wenn *Unique* ungleich 0 ist, wird nicht versucht, die Datei zu erzeugen und zu öffnen.

GetTextAlign

Syntax

```
WORD GetTextAlign(HDC DC);
```

Beschreibung Diese Funktion liefert den Status der Textausrichtungsflags. Die Textausrichtungsflags bestimmen, wie die beiden Funktionen *TextOut* und *ExtTextOut* einen Textstring relativ zu seinem Ursprung ausrichten.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert gibt den Status der Textausrichtungsflags an. Der Rückgabewert ist eine Kombination aus einem oder mehreren der folgenden Werte:

Wert	Bedeutung
TA_BASELINE	Bestimmt die Ausrichtung der x-Achse der ausgewählten Schrift an der Grundlinie des umgebenden Rechtecks.
TA_BOTTOM	Bestimmt die Ausrichtung der x-Achse zur Unterseite des umgebenden Rechtecks.
TA_CENTER	Bestimmt die Ausrichtung der y-Achse zum Mittelpunkt des umgebenden Rechtecks.
TA_LEFT	Bestimmt die Ausrichtung der y-Achse zur linken Seite des umgebenden Rechtecks.
TA_NOUPDATECP	Bestimmt, daß die augenblickliche Position nicht aktualisiert ist.
TA_RIGHT	Bestimmt die Ausrichtung der y-Achse zur rechten Seite des umgebenden Rechtecks.
TA_TOP	Bestimmt die Ausrichtung der x-Achse zur Oberseite des umgebenden Rechtecks.
TA_UPDATECP	Bestimmt, daß die augenblickliche Position aktualisiert ist.

Hinweis Textausrichtungsflags sind nicht unbedingt Einbitflags und können gleich 0 sein. Um zu überprüfen, ob ein einzelnes Flag im Rückgabewert dieser Funktion gesetzt ist, muß man eine Anwendung erstellen, die folgende Schritte ausführt:

1. Anwendung des bitweisen ODER-Operators auf das Flag und seine verknüpften Flags.
   Die folgende Liste zeigt die Gruppen verwandter Flags:

TA_LEFT, *TA_CENTER* und *TA_RIGHT*
TA_BASELINE, *TA_BOTTOM* und *TA_TOP*
TA_NOUPDATECP und *TA_UPDATECP*

2. Anwendung des bitweisen UND-Operators (&) auf das Ergebnis und den Rückgabewert.

3. Test auf Gleichheit dieses Ergebnisses und des Flags.

Das folgende Beispiel zeigt eine Methode, um zu bestimmen, welches horizontale Ausrichtungsflag gesetzt ist:

```
switch (TA_LEFT | TA_RIGHT | TA_CENTER)
& GetTextAlign(DC)) {
  case  TA_LEFT
  .
  .
  .
  case  TA_RIGHT
  .
  .
  .
  case  TA_CENTER
  .
  .
  .
  }
```

Querverweis *TextOut, ExtTextOut*

GetTextCharacterExtra

Syntax `int GetTextCharacterExtra(HDC DC);`

Beschreibung Diese Funktion liefert den aktuellen Zeichenzwischenraum. Der Zeichenzwischenraum definiert den zusätzlichen Raum (in virtuellen Einheiten), den die Funktionen *TextOut* und *ExtTextOut* an jedes Zeichen anhängen, wenn sie eine Zeile schreiben. Das Einfügen von Leerraum dient zum Dehnen von Textzeilen. Ist der aktuelle Abbildungsmodus nicht *MM_TEXT*, formt *GetTextCharacterExtra* das Ergebnis um und rundet es auf die nächste Einheit.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert gibt den aktuellen Zeichenzwischenraum an.

Querverweis *TextOut, ExtTextOut*

GetTextColor

Syntax `DWORD GetTextColor(HDC DC);`

Beschreibung Diese Funktion liefert die aktuelle Textfarbe. Die Textfarbe definiert die Vordergrundfarbe der Zeichen, die mit einer der

beiden Funktionen *TextOut* oder *ExtTextOut* dargestellt werden.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert gibt die aktuelle Textfarbe als RGB-Farbwert an.

Querverweis *TextOut, ExtTextOut*

GetTextExtent

Syntax
```
DWORD GetTextExtent(HDC DC, LPSTR Str, int Count);
```

Beschreibung Diese Funktion berechnet Breite und Höhe der Textzeile, auf die der Parameter *Str* zeigt. *GetTextExtent* verwendet die aktuell ausgewählte Schrift, um die Abmessungen des Strings zu berechnen. Breite und Höhe werden ohne Berücksichtigung der aktuellen Clipping-Region in virtuellen Einheiten berechnet.

DC: Bezeichnet den Gerätekontext.

Str: Zeigt auf einen Textstring.

Count: Bestimmt die Anzahl von Zeichen im Textstring.

Rückgabewert Der Rückgabewert gibt die Abmessungen des Strings an. Die Höhe liegt im niederwertigen, die Breite im höherwertigen WORD.

Hinweis Da einige Geräte Zeichen nicht in regulären Zellen abbilden (das heißt, sie führen *Kerning* aus), kann die Summe der Zeichenabmessungen in einem String eventuell von den Abmessungen des Strings abweichen.

GetTextFace

Syntax
```
int GetTextFace(HDC DC, int Count, LPSTR Facename);
```

Beschreibung Diese Funktion kopiert die Schriftartbezeichnung der ausgewählten Schrift (beispielsweise *System*) in einen Puffer, auf den der Parameter *Facename* zeigt. Die Schriftartenbezeichnung wird als nullterminierter String kopiert. Der Parameter *Count* gibt die maximale Anzahl von Zeichen an, die kopiert werden sollen. Wenn die Schriftartbezeichnung länger ist als die von *Count* vorgegebene Zahl, wird sie abgeschnitten.

DC: Bezeichnet den Gerätekontext.

Count: Bestimmt die Größe des Puffers in Byte (mit Nullzeichen).

Facename: Zeigt auf den Puffer, der die Schriftartbezeichnung aufnehmen soll.

Rückgabewert Der Rückgabewert gibt die Anzahl der tatsächlich in den Puffer kopierten Zeichen an (ohne Nullzeichen). Er ist 0, wenn ein Fehler auftritt.

GetTextMetrics

Syntax
```
BOOL GetTextMetrics(HDC DC, LPTEXTMETRIC Metrics);
```

Beschreibung Diese Funktion füllt die *TEXTMETRIC*-Struktur, auf die *Metrics* zeigt, mit den Abmessungen der ausgewählten Schrift.

DC: Bezeichnet den Gerätekontext.

Metrics: Zeigt auf eine *TEXTMETRIC*-Datenstruktur, die die Abmessungen aufnehmen soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

GetThresholdEvent

Syntax
```
LPINT GetThresholdEvent(void);
```

Beschreibung Diese Funktion liefert ein Flag, das ein kürzlich eingetretenes Schwellenereignis bezeichnet. Ein Schwellenereignis ist jeder Übergang einer Ton-Warteschlange von *n* auf *n-1*, wobei *n* der Schwellenwert in Tönen ist.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert ist ein Zeiger auf ein Schwellenereignis.

GetThresholdStatus

Syntax
```
int GetThresholdStatus(void);
```

Beschreibung Diese Funktion liefert den Schwellenereignisstatus für jeden Ton. Jedes Bit im Status stellt einen Ton dar. Wenn ein Bit gesetzt ist, liegt der Ton-Warteschlangenwert gegenwärtig unter der Schwelle. *GetThresholdStatus* löscht auch das Schwellenereignisflag. Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert gibt die Statusflags des aktuellen Schwellenereignisses an.

GetTickCount

Syntax
```
DWORD GetTickCount(void);
```

Beschreibung Diese Funktion ermittelt die Anzahl von Millisekunden, die vergangen sind, seit das System gestartet wurde. Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert gibt die Anzahl von Millisekunden an, die vergangen sind, seit das System gestartet wurde.

Hinweis Dieser Wert ist auf ca. 55 Millisekunden (+/-) genau.

GetTopWindow

Syntax
```
HWND GetTopWindow(HWND Wnd);
```

Beschreibung Diese Funktion sucht nach einem Handle zum obersten untergeordneten Fenster, das zu dem übergeordneten Fenster *Wnd* gehört. Hat das Fenster keine untergeordneten Fenster, gibt die Funktion 0 zurück.

Wnd: Bezeichnet das übergeordnete Fenster.

Rückgabewert Der Rückgabewert bezeichnet ein Handle zu dem obersten untergeordneten Fenster in der verketteten Liste der untergeordneten Fenster eines übergeordneten Fensters. Wenn kein untergeordnetes Fenster existiert, ist er 0.

GetUpdateRect

Syntax
```
BOOL GetUpdateRect(HWND Wnd, LPRECT Rect, BOOL Erase);
```

Beschreibung Diese Funktion liefert die Koordinaten des kleinsten Rechtecks, das die Aktualisierungsregion des angegebenen Fensters vollständig umschließt.

Wenn das Fenster mit dem Stil *CS_OWNDC* erstellt wurde und der Abbildungsmodus nicht *MM_TEXT* ist, bestimmt *GetUpdateRect* das Rechteck in virtuellen Koordinaten, andernfalls in Client-Koordinaten. Gibt es keine Aktualisierungsregion, leert *GetUpdateRect* das Rechteck (d.h. setzt alle Koordinaten auf 0).

Der Parameter *Erase* gibt an, ob *GetUpdateRect* den Hintergrund der Aktualisierungsregion löschen soll. Wenn *Erase* ungleich Null und die Aktualisierungsregion nicht leer ist, wird der Hintergrund gelöscht. Hierzu sendet *GetUpdateRect* eine *WM_ERASEBKGND*-Botschaft an das angegebene Fenster.

Wnd: Bezeichnet das Fenster, dessen Aktualisierungsregion abgerufen werden soll.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die Client-Koordinaten des umschließenden Rechtecks aufnehmen soll.

Erase: Bestimmt, ob der Hintergrund in der Aktualisierungsregion gelöscht werden soll.

Rückgabewert Der Rückgabewert gibt den Status der Aktualisierungsregion des angegebenen Fensters an. Er ist ungleich Null, wenn die Aktualisierungsregion nicht leer ist, andernfalls ist er Null.

Hinweis Das Aktualisierungsrechteck, das von der Funktion *BeginPaint* abgerufen wird, ist identisch mit dem, das von der Funktion *GetUpdateRect* abgerufen wird.

BeginPaint prüft die Aktualisierungsregion automatisch auf Gültigkeit, so daß ein Aufruf von *GetUpdateRect* unmittelbar nach dem Aufruf von *BeginPaint* eine leere Aktualisierungsregion liefert.

Querverweis *WM_ERASEBKGND*

GetUpdateRgn

Syntax
```
int GetUpdateRgn(HWND Wnd, HRGN Rgn, BOOL Erase);
```

Beschreibung Diese Funktion kopiert die Aktualisierungsregion eines Fensters in die durch den Parameter *Rgn* bezeichnete Region. Die Koordinaten dieser Region werden relativ zur oberen linken Ecke des Fensters (in Client-Koordinaten) gebildet.

Wnd: Bezeichnet das Fenster, das die Aktualisierungsregion enthält.

Rgn: Bezeichnet die Aktualisierungsregion.

Erase: Bestimmt, ob der Fensterhintergrund gelöscht werden soll und ob Nicht-Client-Bereiche von untergeordneten Fenstern gezeichnet werden sollen. Wenn er Null ist, wird nicht gezeichnet.

Rückgabewert Der Rückgabewert gibt den Typ der resultierenden Region an. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
COMPLEXREGION	Die Region hat überlappende Grenzen.
ERROR	Es wurde keine Region erzeugt.
NULLREGION	Die Region ist leer.
SIMPLEREGION	Die Region hat keine überlappenden Grenzen.

Hinweis *BeginPaint* prüft die Aktualisierungsregion automatisch auf Gültigkeit, so daß ein Aufruf von *GetUpdateRgn* unmittelbar nach dem Aufruf von *BeginPaint* eine leere Aktualisierungsregion liefert.

GetVersion

Syntax
```
WORD GetVersion(void);
```

Beschreibung Diese Funktion ermittelt die aktuelle Versionsnummer von Windows. Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert gibt die Haupt- und Nebenversionsnummer von Windows an. Das höherwertige WORD enthält die Nebenversionsnummer (Revision), das niederwertige die Hauptversionsnummer.

GetViewportExt

Syntax
```
DWORD GetViewportExt(HDC DC);
```

Beschreibung Diese Funktion liefert die x- und y-Abmessungen des Grafikfensters des Gerätekontexts.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert gibt die x- und y-Abmessungen in gerätebezogenen Einheiten an. Die y-Abmessung liegt im höherwertigen, die x-Abmessung im niederwertigen WORD.

GetViewportOrg

Syntax
```
DWORD GetViewportOrg(HDC DC);
```

Beschreibung Diese Funktion liefert die x- und y-Koordinaten des Ursprungs des Grafikfensters, das dem angegebenen Gerätekontext zugeordnet ist.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert zeigt den Ursprung des Grafikfensters in gerätebezogenen Koordinaten an. Die y-Koordinate liegt im höherwertigen, die x-Koordinate liegt im niederwertigen WORD.

GetWindow

Syntax
```
HWND GetWindow(HWND Wnd, WORD Cmd);
```

Beschreibung Diese Funktion sucht nach einem Handle zu einem Fenster aus der Liste des Fenster-Managers. Die Liste des Fenster-Managers enthält Einträge für alle Hauptfenster, ihre zugehörenden untergeordneten Fenster sowie deren untergeordnete Fenster. Der Parameter *Cmd* gibt die Beziehung zwischen dem vom Parameter *Wnd* bezeichneten Fenster und dem Fenster, dessen Handle zurückgegeben wird, an.

Wnd: Bezeichnet das ursprüngliche Fenster.

Cmd: Gibt die Beziehung zwischen dem ursprünglichen Fenster und dem zurückgegebenen an. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
GW_CHILD	Bezeichnet das erste untergeordnete Fenster des Fensters.
GW_HWNDFIRST	Gibt das erste Nachbarfenster eines untergeordneten Fensters an. Existiert kein solches Fenster, wird das erste Hauptfenster in der Liste angezeigt.
GW_HWNDLAST	Gibt das letzte Nachbarfenster für ein untergeordnetes Fenster oder das letzte Hauptfenster in der Liste an.
GW_HWNDNEXT	Gibt das Fenster an, das dem vorgegebenen Fenster in der Liste des Fenster-Managers folgt.
GW_HWNDPREV	Gibt das vorhergehende Fenster in der Liste des Fenster-Managers an.
GW_OWNER	Bezeichnet den Besitzer des Fensters.

Rückgabewert Der Rückgabewert bezeichnet ein Fenster. Er ist 0, wenn die Funktion das Ende der Liste des Fenster-Managers erreicht hat oder wenn der Parameter *Cmd* ungültig ist.

GetWindowDC

Syntax
```
HDC GetWindowDC(HWND Wnd);
```

Beschreibung Diese Funktion liefert den Bildschirmkontext für das gesamte Fenster, einschließlich Titelzeile, Menüs und Bildlaufleisten. Ein Fenster-Bildschirmkontext erlaubt es, irgendwo in einem Fenster zu zeichnen – also auch in der Titelzeile, den Menüs und Bildlaufleisten – da der Ursprung des Kontexts die linke obere Ecke des Fensters ist, nicht die des Client-Bereichs.

GetWindowDC weist dem Bildschirmkontext immer, wenn sie den Kontext abruft, Standardattribute zu. Vorherige Attribute gehen verloren.

Wnd: Bezeichnet das Fenster, dessen Bildschirmkontext abgerufen werden soll.

Rückgabewert Der Rückgabewert bezeichnet den Bildschirmkontext für das angegebene Fenster, wenn die Funktion erfolgreich war, andernfalls ist er 0.

Hinweis Die Funktion *GetWindowDC* ist für bestimmte Zeicheneffekte innerhalb des Nicht-Client-Bereichs eines Fensters vorgesehen. Das Zeichnen in Nicht-Client-Bereichen ist jedoch nicht empfehlenswert.

Die Funktion *GetSystemMetrics* kann dazu verwendet werden, die Abmessungen verschiedener Teile des Nicht-Client-

Bereichs, wie zum Beispiel der Titelzeile, des Menüs und der Bildlaufleisten, abzurufen.

Nachdem das Zeichnen beendet ist, muß die Funktion *ReleaseDC* aufgerufen werden, um den Bildschirmkontext freizugeben. Wird er nicht freigegeben, hat dies fatale Auswirkungen auf Zeichenvorgänge, die in anderen Anwendungen erforderlich sind.

Querverweis *GetDC*, *ReleaseDC*

GetWindowExt

Syntax
```
DWORD GetWindowExt(HDC DC);
```

Beschreibung Diese Funktion liefert die x- und y-Abmessungen des Fensters, das dem angegebenen Gerätekontext zugeordnet ist.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert gibt die x- und y-Abmessungen in virtuellen Einheiten an. Die y-Abmessung liegt im höherwertigen, die x-Abmessung im niederwertigen WORD.

GetWindowLong

Syntax
```
LONG GetWindowLong(HWND Wnd, int Index);
```

Beschreibung Diese Funktion liefert Informationen über das vom Parameter *Wnd* bezeichnete Fenster.

Wnd: Bezeichnet das Fenster.

Index: Gibt den Byte-Offset des abzurufenden Wertes an. Es kann auch einer der folgenden Werte sein:

Wert:	Bedeutung
GWL_EXSTYLE	Erweiterter Fensterstil.
GWL_STYLE	Fensterstil.
GWL_WNDPROC	Zeiger auf die Fensterfunktion.

Rückgabewert Der Rückgabewert liefert Informationen über das angegebene Fenster.

Hinweis Um auf alle zusätzlichen Vier-Byte-Werte zuzugreifen, die beim Erzeugen der Fensterklassenstruktur belegt wurden, verwendet man einen positiven Byte-Offset ab dem vom Parameter *Index* bezeichneten Index. Dabei wird mit 0 für den ersten Vier-Byte-Wert im zusätzlichen Speicherraum begonnen, 4 für den nächsten Vier-Byte-Wert, und so weiter.

GetWindowOrg

Syntax
```
DWORD GetWindowOrg(HDC DC);
```

Beschreibung Diese Funktion liefert die x- und y-Koordinaten des Ursprungs des Fensters, das dem angegebenen Gerätekontext zugeordnet ist.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert gibt den Ursprung des Fensters in virtuellen Koordinaten an. Die y-Koordinate liegt im höherwertigen, die x-Koordinate im niederwertigen WORD.

GetWindowRect

Syntax
```
void GetWindowRect(HWND Wnd, LPRECT Rect);
```

Beschreibung Diese Funktion kopiert die Abmessungen des umgrenzenden Rechtecks des angegebenen Fensters in die Struktur, auf die der Parameter *Rect* zeigt.

Die Abmessungen sind in Bildschirmkoordinaten relativ zur oberen linken Ecke des Bildschirms angegeben und schließen die Kopfzeile, die Umrandung und die Bildlaufleisten – falls diese vorhanden sind – ein.

Wnd: Bezeichnet das Fenster.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die Bildschirmkoordinaten der oberen linken und unteren rechten Ecke des Fensters enthält.

Rückgabewert Keiner.

GetWindowsDirectory

Syntax
```
WORD GetWindowsDirectory(LPSTR Buffer, WORD Size);
```

Beschreibung Diese Funktion ermittelt den Pfadnamen des Windows-Verzeichnisses. Das Windows-Verzeichnis enthält unter anderem Windows-Anwendungen, Initialisierungsdateien und Hilfedateien.

Buffer: Zeigt auf den Puffer, der den nullterminierten String aufnehmen soll, der den Pfadnamen enthält.

Size: Bestimmt die maximale Größe des Puffers in Bytes. Dieser Wert sollte auf mindestens 144 gesetzt werden, um genügend Platz für den Pfadnamen im Puffer bereitzustellen.

Rückgabewert Der Rückgabewert ist die Länge des Strings, der in *Buffer* kopiert wurde, ohne Berücksichtigung der abschließenden Null.

Wenn der Rückgabewert größer als *Size* ist, entspricht er der Größe des Puffers, die erforderlich ist, um den Pfadnamen aufzunehmen. Der Rückgabewert ist 0, wenn die Funktion fehlschlägt.

Hinweis

Der von dieser Funktion abgerufene Pfadname endet nicht mit einem Backslash (\), wenn das Windows-Verzeichnis nicht das Hauptverzeichnis des Laufwerks ist. Wenn z.B. das Windows-Verzeichnis auf Laufwerk C: mit *WINDOWS* benannt wurde, so lautet der Pfadname des von der Funktion abgerufenen Windows-Verzeichnisses *C:\WINDOWS*. Wenn Windows im Stammverzeichnis von Laufwerk C: installiert ist, ist der von dieser Funktion abgerufene Pfadname *C:*.

GetWindowTask

Syntax

```
HANDLE GetWindowTask(HWND Wnd);
```

Beschreibung

Diese Funktion liefert das Handle einer Task, die dem Parameter *Wnd* zugeordnet ist. Eine Task ist jedes Programm, das als unabhängige Einheit arbeitet. Alle Anwendungen werden als Task ausgeführt. Jede Instanz einer Anwendung ist eine Task.

Wnd: Bezeichnet das Fenster, für das ein Task-Handle abgerufen wird.

Rückgabewert

Der Rückgabewert bezeichnet die Task, die einem bestimmten Fenster zugeordnet ist.

GetWindowText

Syntax

```
int GetWindowText(HWND Wnd, LPSTR Str, int MaxCount);
```

Beschreibung

Diese Funktion kopiert die Überschrift des angegebenen Fensters in den Puffer, auf den der Parameter *Str* zeigt. Bezeichnet der Parameter *Wnd* ein Steuerelement, kopiert *GetWindowText* den Text des Steuerelements und nicht die Überschrift.

Wnd: Bezeichnet das Fenster oder das Steuerelement, dessen Überschrift bzw. Text kopiert werden soll.

Str: Zeigt auf den Puffer, der den kopierten String aufnehmen soll.

MaxCount: Bestimmt die maximale Anzahl von Zeichen, die in den Puffer kopiert werden sollen (mit Nullzeichen). Wenn der String länger ist als die vom Parameter *MaxCount* vorgegebene Anzahl, wird er abgeschnitten.

Rückgabewert

Der Rückgabewert gibt die Länge des kopierten Strings an (ohne Nullzeichen). Er ist Null, wenn das Fenster keine Überschrift hat oder wenn die Kopfzeile leer ist.

Hinweis Diese Funktion bewirkt, daß eine Botschaft *WM_GETTEXT* an das Fenster oder Steuerelement gesendet wird.

GetWindowTextLength

Syntax
```
int GetWindowTextLength(HWND Wnd);
```

Beschreibung Diese Funktion gibt die Länge der Überschrift des angegebenen Fensters zurück.

Wenn der Parameter *Wnd* ein Steuerelement bezeichnet, gibt die Funktion die Länge des Textes innerhalb des Steuerelements zurück.

Wnd: Bezeichnet das Fenster oder Steuerelement.

Rückgabewert Der Rückgabewert gibt die Textlänge an (ohne Nullzeichen). Er ist Null, wenn kein entsprechender Text existiert.

GetWindowWord

Syntax
```
WORD GetWindowWord(HWND Wnd, int Index);
```

Beschreibung Diese Funktion liefert Informationen über das durch *Wnd* bezeichnete Fenster.

Wnd: Bezeichnet das Fenster.

Index: Bestimmt den Byte-Offset des abzurufenden Werts. Er kann auch einen der folgenden Werte annehmen:

Wert	Bedeutung
GWW_HINSTANCE	Instanz-Handle des Moduls, das das Fenster besitzt.
GWW_HWNDPARENT	Handle des übergeordneten Fensters, wenn es existiert. Die Funktion *SetParent* ändert das übergeordnete Fenster eines untergeordneten Fensters. Eine Anwendung sollte die Funktion *SetWindowLong* nicht aufrufen, um das übergeordnete Fenster eines untergeordneten Fensters zu verändern.
GWW_ID	Steuerelementbezeichner des untergeordneten Fensters.

Rückgabewert Der Rückgabewert enthält Informationen über das bezeichnete Fenster.

Hinweis Um auf zusätzliche Zwei-Byte-Werte zuzugreifen, die beim Erzeugen der Fensterklassenstruktur belegt wurden, benutzt man einen positiven Byte-Offset ab dem vom Parameter *Index* bezeichneten Index – beginnend bei 0 für den ersten Zwei-

Byte-Wert im zusätzlichen Speicher, 2 für den nächsten Zwei-Byte-Wert, und so weiter.

GetWinFlags

Syntax

```
LONG GetWinFlags(void);
```

Beschreibung Diese Funktion gibt einen 32-Bit-Wert zurück, der die Flags enthält, welche die Speicherkonfiguration bezeichnen, unter der Windows läuft. Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert enthält Flags, welche die aktuelle Speicherkonfiguration angeben. Die ODER-Verfügung folgender Werte ist möglich:

Wert	Bedeutung
WF_80X87	System enthält einen mathematischen Intel-Coprozessor.
WF_CPU086	System-CPU ist ein 8086.
WF_CPU186	System-CPU ist ein 80186.
WF_CPU286	System-CPU ist ein 80286.
WF_CPU386	System-CPU ist ein 80386.
WF_CPU486	System-CPU ist ein 80486.
WF_ENHANCED	Windows läuft im erweiterten 386-Modus. Das Flag *WF_PMODE* ist immer gesetzt, wenn *WF_ENHANCED* gesetzt ist.
WF_LARGEFRAME	Windows läuft mit Large-Frame-EMS-Speicherkonfiguration.
WF_PMODE	Windows läuft im Protected Mode. Dieses Flag ist immer gesetzt, wenn entweder *WF_ENHANCED* oder *WF_STANDARD* gesetzt ist.
WF_SMALLFRAME	Windows läuft mit Small-Frame-EMS- Speicherkonfiguration.
WF_STANDARD	Windows läuft im Standardmodus. Das Flag *WF_PMODE* ist immer gesetzt, wenn *WF_STANDARD* gesetzt ist Wenn weder *WF_ENHANCED* noch *WF_STANDARD* gesetzt sind, läuft Windows im Real Mode.

GlobalAddAtom

Syntax

```
ATOM GlobalAddAtom(LPSTR Str);
```

Beschreibung Diese Funktion fügt den String, auf den der Parameter *Str* zeigt, der Atomtabelle hinzu und erzeugt ein neues globales Atom, das den String eindeutig bezeichnet.

Ein globales Atom ist ein Atom, das für alle Anwendungen verfügbar ist. Das Atom kann in einem Aufruf der nachfolgenden Funktion *GlobalGetAtomName* benutzt werden, um den String aus der Atomtabelle abzurufen.

GlobalAddAtom speichert nur eine Kopie des gegebenen Strings in der Atomtabelle. Existiert der String bereits in der Tabelle, gibt die Funktion den existierenden Atomwert zurück und erhöht den Referenzzähler des Strings um 1. Der Referenzzähler des Strings ist eine Zahl, die angibt, wie oft *GlobalAddAtom* für einen bestimmten String aufgerufen wurde.

Str: Zeigt auf den String, der zur Atomtabelle hinzugefügt werden soll. Der String muß ein nullterminierter String sein.

Rückgabewert Der Rückgabewert bezeichnet das neu erzeugte Atom, wenn die Funktion erfolgreich war. Andernfalls ist er 0.

Hinweis Die von *GlobalAddAtom* zurückgegebenen Atomwerte liegen innerhalb des Bereiches von 0xC000 bis 0xFFFF.

GlobalAlloc

Syntax
```
HANDLE GlobalAlloc(WORD Flags, DWORD Bytes);
```

Beschreibung Diese Funktion reserviert die vom Parameter *Bytes* vorgegebene Zahl von Bytes im globalen Heap. Der Speicher kann fest oder verschiebbar sein, abhängig von dem durch den Parameter *Flags* angegebenen Speichertyp.

Flags: Bestimmt ein oder mehrere Flags, die der Funktion *GlobalAlloc* mitteilen, wie der Speicher reserviert werden soll. Folgende Werte sind möglich:

Wert	Bedeutung
GMEM_DDESHARE	Reserviert gemeinsam nutzbaren Speicher. Wird nur für dynamischen Datenaustausch (DDE) verwendet. Es ist jedoch zu beachten, daß Windows automatisch Speicher verwirft, der mit diesem Attribut reserviert wurde, wenn die Anwendung, die ihn reserviert hat, endet
GMEM_DISCARDABLE	Reserviert Speicher und ordnet ihm das Attribut »discardable« zu. Kann nur mit *GMEM_MOVEABLE* benutzt werden.
GMEM_FIXED	Reserviert festen Speicher.
GMEM_MOVEABLE	Reserviert verschiebbaren Speicher. Kann nicht mit *GMEM_FIXED* benutzt werden.
GMEM_NOCOMPACT	Verdichtet und verwirft nicht, um den Belegungsbedarf zu befriedigen.

Wert	Bedeutung
GMEM_NODISCARD	Verwirft nicht, um den Belegungsbedarf zu befriedigen.
GMEM_NOT_BANKED	Belegt Speicher, der nicht in Bänke aufgeteilt ist. Kann nicht mit *GMEM_NOTIFY* benutzt werden.
GMEM_NOTIFY	Ruft die Benachrichtigungsroutine auf, falls das Speicherobjekt verworfen wird.
GMEM_ZEROINIT	Initialisiert Speicherinhalte mit 0.

Nach Auswahl von *GMEM_FIXED* oder *GMEM_MOVEABLE* Können weitere Flags mit dem bitweisen ODER-Operator (|) kombiniert werden.

Bytes: Bestimmt die Zahl von Bytes, die reserviert werden sollen.

Rückgabewert Der Rückgabewert bezeichnet den reservierten globalen Speicher, wenn die Funktion erfolgreich war. Andernfalls ist er 0.

Hinweis Wenn die Funktion erfolgreich war, reserviert sie mindestens den angeforderten Speicherplatz. Der tatsächlich reservierte Speicherplatz kann größer sein und die Anwendung kann den gesamten Bereich nutzen. Um die tatsächlich reservierte Speichermenge zu bestimmen, rufen Sie die Funktion *GlobalSize* auf.

Der größte Speicherblock, den eine Anwendung belegen kann, umfaßt 1 MByte im Standardmodus und 64 MByte im erweiterten 386-Modus.

Querverweis *GlobalSize*

GlobalCompact

Syntax
```
DWORD GlobalCompact(DWORD MinFree);
```

Beschreibung Diese Funktion erzeugt die Anzahl freier Bytes im globalen Speicher, die der Parameter *MinFree* bezeichnet, indem sie den globalen Heap des Systems verdichtet und gegebenenfalls Teile des Heaps verwirft.

Die Funktion verdichtet immer zuerst den Speicher, bevor sie nach freien Speicherbereichen sucht. Sie überprüft dann den globalen Speicher auf einen Bereich aufeinanderfolgender freier Bytes, in der Größe, die der Parameter *MinFree* vorgibt. Wenn kein solcher Bereich existiert, verwirft die Funktion *GlobalCompact* ungesperrte und verwerfbare Blöcke, bis der erforderliche Platz erzeugt ist.

MinFree: Bestimmt die gewünschte Anzahl freier Bytes.

Rückgabewert Der Rückgabewert gibt die Anzahl von Bytes im größten Block freien Speichers an.

Hinweis Wenn *MinFree* 0 ist, gibt der Rückgabewert die Zahl von Bytes im größten freien Segment an, das Windows erzeugen kann, wenn es alle verwerfbaren Segmente löscht.

Wenn eine Anwendung den Rückgabewert als Parameter *Bytes* für die Funktion *GlobalAlloc* benutzt, dann sollten die Flags *GMEM_NOCOMPACT* und *GMEM_NODISCARD* nicht verwendet werden.

GlobalDeleteAtom

Syntax
```
ATOM GlobalDeleteAtom(ATOM AnAtom);
```

Beschreibung Diese Funktion verringert den Referenzzähler eines globalen Atoms um 1. Wenn der Referenzzähler des Atoms 0 wird, löscht die Funktion den zugeordneten String aus der Atomtabelle. Ein globales Atom ist ein Atom, das für alle Windows-Anwendungen verfügbar ist.

Der Referenzzähler eines Atoms gibt an, wie oft das Atom zur Atomtabelle hinzugefügt wurde. Die Funktion *GlobalAddAtom* erhöht den Zähler bei jedem Aufruf und die Funktion *GlobalDeleteAtom* löscht den String nur, wenn der Referenzzähler des Atoms 0 ist.

AnAtom: Bezeichnet das Atom und den String, die gelöscht werden sollen.

Rückgabewert Der Rückgabewert gibt das Ergebnis der Funktion an. Er ist 0, wenn die Funktion erfolgreich war. Er ist gleich *AnAtom*, wenn die Funktion fehlschlägt und das Atom nicht gelöscht wurde.

GlobalDiscard

Syntax
```
HANDLE GlobalDiscard(HANDLE Mem);
```

Beschreibung Dieses Makro verwirft einen globalen Speicherblock, der vom Parameter *Mem* bezeichnet wird. Der Sperrenzähler muß 0 sein.

Der globale Speicherblock wird aus dem Speicher gelöscht, sein Handle bleibt aber gültig. Eine Anwendung kann nachfolgend das Handle an die Funktion *GlobalReAlloc* übergeben, um einen anderen globalen Speicherblock, der vom gleichen Handle bezeichnet wird, zu belegen.

Mem: Bezeichnet den globalen Speicherblock, der verworfen werden soll.

Rückgabewert Wenn die Funktion erfolgreich war, bezeichnet der Rückgabewert den verworfenen Block, andernfalls ist er 0.

Hinweis Die Funktion verwirft nur globale Objekte, die von einer Anwendung mit gesetzten Flags *GMEM_DISCARDABLE* und *GMEM_MOVEABLE* zugewiesen wurden. Sie kann nicht ausgeführt werden, wenn eine Anwendung versucht, ein festes oder gesperrtes Objekt zu verwerfen.

GlobalDosAlloc

Syntax
```
DWORD GlobalDosAlloc(DWORD Bytes)
```

Beschreibung Diese Funktion belegt globalen Speicher, auf den von DOS im Real Mode zugegriffen werden kann. Es wird garantiert, daß der Speicher im ersten MByte des linearen Adressraums liegt.

Bytes: Bestimmt die Anzahl der zu reservierenden Bytes.

Rückgabewert Der Rückgabewert enthält einen Paragraphsegmentwert in seinem höherwertigen und einen Selektor in seinem niederwertigen Wort. Eine Anwendung kann den Paragraphsegmentwert verwenden, um im Real Mode und den Selektor, um im Protected Mode auf den Speicher zuzugreifen. Läuft Windows im Real Mode, so sind das höherwertige und das niederwertige Wort gleich. Wenn Windows keinen Speicherblock mit der geforderten Größe reservieren kann, so wird NULL zurückgeliefert.

Hinweis Eine Anwendung sollte diese Funktion nur dann benutzen, wenn es unbedingt notwendig ist.

GlobalDosFree

Syntax
```
DWORD GlobalDosFree(DWORD Selector);
```

Beschreibung Diese Funktion gibt einen globalen Speicherblock frei, der vorher durch einen Aufruf der Funktion *GlobalDosAlloc* reserviert wurde.

Selector: Bestimmt den Speicher, der freigegeben werden soll.

Rückgabewert Bei erfolgreicher Ausführung wird NULL zurückgeliefert, andernfalls ist der Rückgabewert gleich *Selector*.

GlobalFindAtom

Syntax
```
ATOM GlobalFindAtom(LPSTR Str);
```

Beschreibung Diese Funktion sucht in der Atomtabelle nach dem String, auf den der Parameter *Str* zeigt, und liefert das globale Atom, das diesem String zugeordnet ist.

Ein globales Atom ist ein Atom, das allen Windows-Anwendungen verfügbar ist.

Str: Zeigt auf den String, nach dem gesucht werden soll. Der String muß nullterminiert sein.

Rückgabewert Der Rückgabewert bezeichnet das globale Atom, das dem gegebenen String zugeordnet ist. Er ist 0, wenn der String nicht in der Tabelle ist.

GlobalFix

Syntax
```
void GlobalFix(HANDLE Mem);
```

Beschreibung Diese Funktion verhindert, daß der vom Parameter *Mem* bezeichnete globale Speicherblock im linearen Speicher verschoben wird.

Der Block wird im linearen Speicher an seiner aktuellen Adresse gesperrt und sein Sperrenzähler um 1 erhöht. Gesperrter Speicher wird nicht verschoben oder verworfen, es sei denn, daß der Speicherblock von der Funktion *GlobalReAlloc* neu zugewiesen wird.

Der Block bleibt solange gesperrt im Speicher, bis sein Sperrenzähler auf 0 zurückgesetzt wurde.

Jedesmal, wenn eine Anwendung *GlobalFix* für ein Speicherobjekt aufgerufen hat, muß sie entsprechend *GlobalUnfix* für das Objekt aufrufen. Die Funktion *GlobalUnfix* verringert den Sperrenzähler für das Objekt.

Andere Funktionen können den Sperrenzähler eines Speicherobjektes ebenfalls beeinflussen. Die Beschreibung von *GlobalFlags* enthält eine Liste der Funktionen, die den Sperrenzähler beeinflussen.

Mem: Bezeichnet den globalen Speicherblock.

Hinweis Der Aufruf dieser Funktion stört die Windows-Speicherverwaltung und führt zu einer Fragmentierung des linearen Adressraums. Nur sehr wenige Anwendungen benötigen Speicher im linearen Adressraum.

GlobalFlags

Syntax
```
WORD GlobalFlags(HANDLE Mem);
```

Beschreibung Diese Funktion gibt Informationen über den globalen Speicherblock zurück, den der Parameter *Mem* bezeichnet.

Mem: Bezeichnet den globalen Speicherblock.

Rückgabewert Der Rückgabewert enthält ein Speicherbelegungsflag im höherwertigen Byte.

Das Flag kann einen der in der folgenden Tabelle beschriebenen Werte annehmen:

Wert	Bedeutung
GMEM_DDESHARE	Der Block kann gemeinsam genutzt werden. Dies wird nur für dynamischen Datenaustausch (DDE) benötigt.
GMEM_DISCARDABLE	Der Block kann verworfen werden.
GMEM_DISCARDED	Der Block wurde verworfen.
GMEM_NOT_BANKED	Der Block kann nicht in Bänke aufgeteilt werden.

Das niederwertige Byte des Rückgabewerts enthält den Sperrenzähler des Blocks. Der Wert des Sperrenzählers kann mit der Maske *GMEM_LOCKCOUNT* aus dem Rückgabewert ermittelt werden.

Hinweis

Um zu testen, ob ein Objekt verworfen werden kann oder nicht, verknüpft man den Rückgabewert von *GlobalFlags* mit *GMEM_DISCARDABLE* über den UND-Operator (&).

Die folgenden Funktionen können den Sperrenzähler eines globalen Speicherblocks beeinflussen:

Erhöht den Sperrenzähler	Verringert den Sperrenzähler
GlobalFix	*GlobalUnfix*
GlobalLock	*GlobalUnlock*
GlobalWire	*GlobalUnWire*
LockSegment	*UnlockSegment*

GlobalFree

Syntax

```
HANDLE GlobalFree(HANDLE Mem);
```

Beschreibung

Diese Funktion gibt den globalen Speicherblock frei, der vom Parameter *Mem* bezeichnet wird und macht das Handle des Speicherblocks ungültig.

Mem: Bezeichnet den globalen Speicherblock, der freigegeben werden soll.

Rückgabewert

Bei erfolgreicher Ausführung wird 0 zurückgegeben, andernfalls ist der Rückgabewert gleich *Mem*.

Hinweis

GlobalFree darf nicht verwendet werden, um einen gesperrten Speicherblock freizugeben, also einen Speicherblock, dessen Sperrenzähler größer 0 ist.

Die Beschreibung der Funktion *GlobalFlags* enthält eine Liste der Funktionen, die den Sperrenzähler beeinflussen.

GlobalGetAtomName

Syntax
```
WORD GlobalGetAtomName(ATOM AnAtom, LPSTR Buffer,
int Size);
```

Beschreibung Diese Funktion liefert eine Kopie des Strings, der dem Parameter *AnAtom* zugeordnet ist, und legt sie in dem Puffer ab, auf den der Parameter *Buffer* zeigt. Der Parameter *Size* bestimmt die maximale Größe des Puffers. Ein globales Atom ist ein Atom, das allen Windows-Anwendungen verfügbar ist.

AnAtom: Bezeichnet den String, der abgerufen werden soll.

Buffer: Zeigt auf den Puffer, der den String aufnehmen soll.

Size: Bestimmt die maximale Größe des Puffers in Bytes (mit Nullzeichen).

Rückgabewert Der Rückgabewert ist die tatsächliche Anzahl der in den Puffer kopierten Bytes (ohne Nullzeichen). Er ist 0, wenn das angegebene Atom ungültig ist.

GlobalHandle

Syntax
```
DWORD GlobalHandle(WORD Mem);
```

Beschreibung Diese Funktion liefert das Handle des globalen Speicherobjekts, dessen Segmentadresse oder Selektor vom Parameter *Mem* bezeichnet wird.

Mem: Bezeichnet die Segmentadresse oder den Selektor eines globalen Speicherobjekts.

Rückgabewert Das niederwertige WORD des Rückgabewerts enthält das Handle des globalen Speicherobjekts, das höherwertige die Segmentadresse oder den Selektor des Speicherobjekts. Existiert für das Speicherobjekt kein Handle, wird 0 zurückgegeben.

GlobalLock

Syntax
```
LPSTR GlobalLock(HANDLE Mem);
```

Beschreibung Diese Funktion liefert einen Zeiger auf den globalen Speicherblock, den der Parameter *Mem* bezeichnet.

Außer bei nichtverwerfbaren Objekten im Protected Mode (Standardmodus oder erweiterter 386-Modus) wird der Block im Speicher an der angegebenen Adresse gesperrt und sein Sperrenzähler um 1 erhöht. Gesperrter Speicher kann nicht verschoben oder verworfen werden. Möglich ist aber die Neubelegung des Speicherblocks durch die Funktion *GlobalReAlloc*. Im Protected Mode erhöht *GlobalLock* den Sperrenzähler nur

bei Objekten mit dem Attribut »discardable« und automatischen Datensegmenten.

Jedes Mal, wenn eine Anwendung *GlobalLock* für ein Objekt aufgerufen hat, muß sie auch *GlobalUnlock* aufrufen. Die Funktion *GlobalUnlock* verringert den Sperrenzähler für das Objekt, wenn *GlobalLock* den Sperrenzähler erhöht hat.

Andere Funktionen können den Sperrenzähler eines Objekts ebenfalls beeinflussen. Die Beschreibung der Funktion *GlobalFlags* enthält eine Liste dieser Funktionen.

Mem: Bezeichnet den globalen Speicherblock, der gesperrt werden soll.

Rückgabewert Bei erfolgreicher Ausführung zeigt der Rückgabewert auf das erste Speicherbyte im globalen Block. Wenn das Objekt verworfen wurde oder ein Fehler auftritt, wird 0 zurückgeliefert.

Hinweis Verworfene Objekte haben den Sperrenzählerwert 0.

GlobalLRUNewest

Syntax
```
HANDLE GlobalLRUNewest(HANDLE Mem);
```

Beschreibung Diese Funktion verschiebt das globale Speicherobjekt, das von *Mem* bezeichnet wird, an die neueste, zuletzt benutzte (least-recently-used, LRU) Position im Speicher. Dies reduziert die Wahrscheinlichkeit, daß das Objekt bald verworfen wird, verhindert aber das Verwerfen nicht.

Mem: Bezeichnet das globale Speicherobjekt, das verschoben werden soll.

Rückgabewert Der Rückgabewert ist 0, wenn der Parameter *Mem* kein gültiges Handle bezeichnet.

Hinweis Diese Funktion ist nur dann hilfreich, wenn das bezeichnete Speicherobjekt das Attribut «discardable« hat.

GlobalLRUOldest

Syntax
```
HANDLE GlobalLRUOldest(HANDLE Mem);
```

Beschreibung Diese Funktion verschiebt das globale Speicherobjekt, das von *Mem* bezeichnet wird, an die älteste zuletzt benutzte (least-recently-used, LRU) Position im Speicher und macht es damit zum nächsten Speicherobjekt, das verworfen wird.

Mem: Bezeichnet das globale Speicherobjekt, das verschoben werden soll.

Rückgabewert Der Rückgabewert ist 0, wenn der Parameter *Mem* kein gültiges Handle bezeichnet.

Hinweis Diese Funktion ist nur dann hilfreich, wenn das bezeichnete Speicherobjekt das Attribut »discardable« hat.

GlobalNotify

Syntax
```
void GlobalNotify(FARPROC NotifyProc);
```

Beschreibung Diese Funktion installiert eine Benachrichtigungsprozedur für die aktuelle Task. Windows ruft diese Prozedur immer dann auf, wenn ein globaler Speicherblock, der mit dem Flag *GMEM_NOTIFY* reserviert wurde, verworfen werden soll.

NotifyProc: Die Adresse der Prozedurinstanz der aktuellen Benachrichtigungsprozedur der Task.

Rückgabewert Keiner.

Hinweis Eine Anwendung darf *GlobalNotify* nur einmal pro Instanz aufrufen.

Windows ruft die Benachrichtigungsprozedur nicht auf, wenn es Speicher verwirft, der zu einer DLL (dynamischen Linkbibliothek) gehört.

Wurde das Objekt verworfen, muß die Anwendung das Flag *GMEM_NOTIFY* verwenden, wenn sie das Objekt durch Aufruf der Funktion *GlobalRealloc* neu erstellt. Andernfalls wird die Anwendung nicht benachrichtigt, wenn das Objekt wieder verworfen wird.

Wenn die Benachrichtigungsprozedur einen Wert ungleich 0 zurückgibt, verwirft Windows den globalen Speicherblock. Wenn sie 0 zurückgibt, wird der Block nicht verworfen.

Die Callback-Funktion muß in einem festen Code-Segment der DLL liegen.

Callback-Funktion
```
BOOL FAR PASCAL NotifyProc(HANDLE Mem);
```

NotifyProc: Ein Platzhalter für den in der Anwendung definierten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der DLL exportiert werden.

Mem: Bezeichnet den globalen Speicherblock, der verworfen wurde.

Rückgabewert Die Funktion gibt einen Wert ungleich 0 zurück, wenn Windows den Speicherblock verwerfen soll und 0, wenn dieser nicht verworfen werden soll.

Hinweis Die Callback-Funktion wird nicht unbedingt im Kontext der Anwendung aufgerufen, die die Routine besitzt. Aus diesem Grund sollte die Callback-Funktion nicht vom Stack-Segment der Anwendung ausgehen. Die Callback-Funktion sollte keine Routine aufrufen, die Speicher verschieben könnte.

GlobalPageLock

Syntax
```
WORD GlobalPageLock(HANDLE Selector);
```

Beschreibung Diese Funktion erhöht den Seitensperrenzähler des Speichers, der dem angegebenen globalen Selektor zugeordnet ist. Solange sein Seitensperrenzähler ungleich 0 ist, bleiben die Daten, auf die sich der Selektor bezieht, im Speicher an der jeweiligen physikalischen Adresse seitenweise gespeichert.

Die Funktion *GlobalPageLock* erhöht den Seitensperrenzähler für den Speicherblock, die Funktion *GlobalPageUnlock* verringert ihn. Seitensperroperationen können verschachtelt sein, aber jede Seitensperrung muß durch eine entsprechende Aufhebung der Sperre ausgeglichen werden.

Selector: Bestimmt den Selektor des Speichers, der seitenweise gesperrt werden soll.

Rückgabewert Der Rückgabewert gibt den Wert des Seitensperrenzählers an, nachdem die Funktion ihn erhöht hat. Im Fall eines Fehlers wird 0 zurückgeliefert.

Hinweis Eine Anwendung sollte diese Funktion nur verwenden, wenn dies unbedingt notwendig ist.

Die Benutzung dieser Funktion verletzt alle gebräuchlichen Windows-Programmierpraktiken. Sie ist zum Gebrauch für dynamisch reservierte Daten vorgesehen, auf die von einem Interrupt zugegriffen werden muß. Aus diesem Grund darf sie auch nur von einer DLL (dynamischen Linkbibliothek) aufgerufen werden.

Querverweis *GlobalPageUnlock*

GlobalPageUnlock

Syntax
```
WORD GlobalPageUnlock(HANDLE Selector);
```

Beschreibung Diese Funktion verringert den Seitensperrenzähler für den vom Parameter *Selector* bezeichneten Speicherblock um eins, und ermöglicht, daß der Speicherblock verschoben und auf Platte ausgelagert werden kann, wenn der Seitensperrenzähler 0 erreicht. *GlobalPageLock* erhöht den Seitensperrenzähler für den Speicherblock, *GlobalPageUnlock* verringert ihn.

Seitensperroperationen können verschachtelt sein, jede Seitensperrung muß jedoch durch eine entsprechende Aufhebung der Sperre ausgeglichen werden. Diese Funktion kann nur von dynamischen Linkbibliotheken (DLL) aufgerufen werden.

Selector: Bestimmt den Selektor des Speichers, dessen Seitensperrenzähler verringert werden soll.

Rückgabewert Bestimmt den Wert des Seitensperrenzählers, nachdem die Funktion ihn verringert hat. Im Fall eines Fehlers wird 0 zurückgegeben.

GlobalReAlloc

Syntax
```
HANDLE GlobalReAlloc(HANDLE Mem, DWORD Bytes,
WORD Flags);
```

Beschreibung Diese Funktion weist den vom Parameter *Mem* bezeichneten globalen Speicherblock neu zu, indem sie seine Größe auf die vom Parameter *Bytes* vorgegebene Zahl von Bytes verringert oder erhöht.

Mem: Bezeichnet den globalen Speicherblock, der neu zugewiesen werden soll.

Bytes: Bestimmt die neue Größe des Speicherblocks.

Flags: Bestimmt, wie der globale Block neu reserviert werden soll. Wenn die vorhandenen Speicherflags geändert werden können, benutzt man entweder einen oder mehrere der folgenden Flags (wenn mehrere Flags angegeben sind, verknüpft man sie mit dem bitweisen ODER-Operator (|)):

Wert	Bedeutung
GMEM_DISCARDABLE	Speicher kann verworfen werden. Nur mit *GMEM_MODIFY* zu benutzen.
GMEM_MODIFY	Speicherflags werden verändert. Der Parameter *Bytes* wird ignoriert. Nur zu benutzen, wenn eine Anwendung vorhandene Speicherflags verändern und den Speicherblock nicht mit einer neuen Größe reservieren soll.
GMEM_MOVEABLE	Speicher ist verschiebbar. Wenn *Bytes* 0 ist, bewirkt dieses Flag, daß ein Objekt, das vorher als verschiebbar und verwerfbar reserviert wurde, verworfen wird, wenn der Sperrenzähler des Blocks 0 ist. Wenn der Block nicht verschiebbar und nicht verwerfbar ist, kann *GlobalReAlloc* nicht ausgeführt werden. Wenn *Bytes* ungleich 0 und der von *Mem* angegebene Block fixiert ist, erlaubt dieses Flag die Verschiebung des neu reservierten Blocks an jede feste Speicherstelle. Wenn ein verschiebbares Objekt gesperrt ist, erlaubt dieses Flag dessen Verschiebung.

Wert	Bedeutung
	Dies ist sogar dann der Fall, wenn das Objekt durch einen unmittelbar vorangegangenen Aufruf von *GlobalLock* gesperrt wurde. Beachten Sie, daß das Handle, welches von *GlobalReAlloc* zurückgegeben wird, sich in diesem Fall von dem Handle unterscheidet, das an die Funktion übergeben wurde. Dieses Flag sollte mit *GMEM_MODIFY* benutzt werden, um einen festen Speicherblock verschiebbar zu machen.
GMEM_NOCOMPACT	Speicher wird nicht verdichtet oder verworfen, um den Belegungsbedarf zu befriedigen. Dieses Flag wird ignoriert, wenn *GMEM_MODIFY* gesetzt ist.
GMEM_NODISCARD	Objekte werden nicht verworfen, um den Belegungsbedarf zu befriedigen. Dieses Flag wird ignoriert, wenn das Flag *GMEM_MODIFY* gesetzt ist.
GMEM_ZEROINIT	Wird der Block vergrößert, werden die zusätzlichen Speicherinhalte mit 0 initialisiert. Dieses Flag wird ignoriert, wenn *GMEM_MODIFY* gesetzt ist.

Rückgabewert Bei erfolgreicher Ausführung bezeichnet der Rückgabewert den neu reservierten globalen Speicher. Wenn der Block nicht neu zugewiesen werden konnte, wird 0 zurückgegeben.

Der Rückgabewert ist nach erfolgreicher Ausführung immer identisch mit dem Parameter *Mem*, wenn nicht eine der folgenden Bedingungen auftritt:

- Das Flag *GMEM_MOVEABLE* wird verwendet, um die Verschiebung eines festen Blocks an eine neue feste Position zu ermöglichen.
- Windows läuft im Standardmodus, und die Neuzuweisung des Objektes überschreitet ein Vielfaches von 65519 Bytes (17 Bytes weniger als 64 KByte).
- Windows läuft im erweiterten 386-Modus, und die Neuzuweisung des Objektes überschreitet ein Vielfaches von 64 KByte.

GlobalSize

Syntax
```
DWORD GlobalSize(HANDLE Mem);
```

Beschreibung Diese Funktion liefert die aktuelle Größe des globalen Speicherblocks, den der Parameter *Mem* bezeichnet, in Bytes.

Mem: Bezeichnet den globalen Speicherblock.

Rückgabewert Der Rückgabewert gibt die tatsächliche Größe des angegebenen Speicherblocks in Bytes an. Er ist 0, wenn das gegebene Handle nicht gültig ist oder wenn das Objekt verworfen wurde.

Hinweis Die tatsächliche Größe eines Speicherblocks kann größer sein als die angeforderte Größe.

Eine Anwendung sollte die Funktion *GlobalFlags* vor *GlobalSize* aufrufen, um zu überprüfen, ob der angegebene Speicherblock nicht verworfen wurde. Ist der Speicherblock bereits verworfen, wäre der Rückgabewert von *GlobalSize* bedeutungslos.

GlobalUnfix

Syntax
```
BOOL GlobalUnfix(HANDLE Mem);
```

Beschreibung Diese Funktion hebt die Sperre des durch den Parameter *Mem* bezeichneten globalen Speicherblocks auf.

GlobalUnfix verringert den Sperrenzähler des Blocks um 1. Wenn der Sperrenzähler bis auf 0 verringert wurde, ist die Sperre des Blocks vollständig aufgehoben und der Block kann verschoben oder verworfen werden.

Andere Funktionen können den Sperrenzähler eines Speicherobjekts ebenfalls beeinflussen. In der Beschreibung der Funktion *GlobalFlags* findet sich eine Liste dieser Funktionen.

Wenn eine Anwendung *GlobalFix* für ein Objekt aufruft, muß sie für dieses auch *GlobalUnfix* aufrufen.

Mem: Bezeichnet den globalen Speicherblock, dessen Sperre aufgehoben werden soll.

Rückgabewert Der Rückgabewert ist Null, wenn der Sperrenzähler des Blocks bis auf 0 verringert wurde, andernfalls ist er ungleich Null.

Querverweis *GlobalFlags*

GlobalUnlock

Syntax
```
BOOL GlobalUnlock(HANDLE Mem);
```

Beschreibung Diese Funktion hebt die Sperre des durch den Parameter *Mem* bezeichneten globalen Speicherblocks auf.

Im Real Mode (oder wenn der Block nicht das Attribut »discardable« hat) verringert *GlobalUnlock* den Sperrenzähler des Blocks um 1. Im Protected Mode verringert *GlobalUnlock* nur bei Objekten mit dem Attribut »discardable« und bei automatischen Datensegmenten den Sperrenzähler. Wenn der Sperrenzähler bis auf 0 verringert wurde, ist die Sperre des Blocks gänzlich aufgehoben und der Block kann verschoben

oder verworfen werden. Die Beschreibung der Funktion *GlobalFlags* enthält eine Liste der Funktionen, die den Sperrenzähler beeinflussen.

Ruft eine Anwendung für ein Objekt *GlobalLock* auf, so muß sie für dieses auch *GlobalUnlock* aufrufen.

Mem: Bezeichnet den globalen Speicherblock, dessen Sperre aufgehoben werden soll.

Rückgabewert Der Rückgabewert bezeichnet das Ergebnis der Funktion. Er ist Null, wenn der Sperrenzähler des Blocks bis auf 0 verringert wurde, andernfalls ist er ungleich Null.

Eine Anwendung sollte sich nicht auf den Rückgabewert verlassen, um zu bestimmen, wie oft sie *GlobalUnlock* für einen Speicherblock aufrufen muß.

GlobalUnWire

Syntax
```
BOOL GlobalUnWire(HANDLE Mem);
```

Beschreibung Diese Funktion hebt die Sperre eines Speichersegments auf, das von der Funktion *GlobalWire* gesperrt wurde, und verringert den Sperrenzähler um 1. Wenn der Sperrenzähler bis auf 0 verringert wurde, ist die Sperre des Segments vollständig aufgehoben und das Segment kann verschoben oder verworfen werden.

Es gibt auch andere Funktionen, die den Sperrenzähler eines Speicherobjekts beeinflussen können. Die Beschreibung der Funktion *GlobalFlags* enthält eine Liste dieser Funktionen.

Ruft eine Anwendung für ein Objekt *GlobalWire* auf, so muß sie für dieses auch *GlobalUnWire* aufrufen.

Mem: Bezeichnet das Segment, dessen Sperre aufgehoben werden soll.

Rückgabewert Wenn die Sperre des Speichersegments aufgehoben, also der Sperrenzähler auf 0 verringert wurde, wird ein Wert ungleich Null, andernfalls Null zurückgeliefert.

GlobalWire

Syntax
```
LPSTR GlobalWire(HANDLE Mem);
```

Beschreibung Diese Funktion verschiebt ein Segment in den unteren Speicherbereich und sperrt es. Sie ist sehr nützlich, wenn eine Anwendung ein Segment über längere Zeit hinweg sperren muß. Das Sperren eines Segments aus dem mittleren Speicherbereich über längere Zeit kann Speicherverwaltungsprobleme verursachen, da dies die Größe des größten zusammenhängenden verfügbaren Speicherblocks reduziert. *GlobalWire* ver-

schiebt ein Segment an die unterste mögliche Adresse im Speicher und sperrt es, wodurch der Speicherbereich, den Windows am häufigsten verwendet, freigegeben wird.

Ruft eine Anwendung für ein Objekt die Funktion *GlobalWire* auf, so muß sie für dieses auch *GlobalUnWire* aufrufen. *GlobalUnWire* verringert den Sperrenzähler für das Objekt. Andere Funktionen können den Sperrenzähler eines Speicherobjekts ebenfalls beeinflussen. Die Beschreibung der Funktion *GlobalFlags* enthält eine Liste dieser Funktionen.

Eine Anwendung darf nicht die Funktion *GlobalUnlock* aufrufen, um die Sperre für das Objekt aufzuheben.

Mem: Bezeichnet das Segment, das verschoben und gesperrt werden soll.

Rückgabewert Der Rückgabewert zeigt auf die neue Segmentposition. Er ist 0, wenn die Funktion fehlgeschlagen ist.

Querverweis *GlobalUnWire*

GrayString

Syntax
```
BOOL GrayString(HDC DC, HBRUSH Brush,
FARPROC OutputFunc, DWORD Data, int Count, int X,
int Y, int Width, int Height);
```

Beschreibung Diese Funktion stellt an der angegebenen Position grauen, nicht aktivierbaren Text dar. Die Funktion *GrayString* stellt den Text dar, indem sie ihn in ein Speicher-Bitmap schreibt, das Bitmap grau färbt und es dann auf den Bildschirm kopiert. Die Funktion färbt den Text ungeachtet des ausgewählten Pinsels und Hintergrunds ein. *GrayString* verwendet die aktuell ausgewählte Schrift für den aktuellen Gerätekontext, den der Parameter *DC* vorgibt.

Wenn der Parameter *OutputFunc* 0 ist, ruft die grafische Geräteschnittstelle (GDI) die Funktion *TextOut* auf. Dabei wird davon ausgegangen, daß der Parameter *Data* ein Zeiger auf den Wert ist, der ausgegeben werden soll. Wenn die Zeichen, die ausgegeben werden sollen, von *TextOut* nicht bearbeitet werden können (weil zum Beispiel der String als Bitmap gespeichert ist), muß die Anwendung eine eigene Ausgabefunktion zur Verfügung stellen.

DC: Bezeichnet den Gerätekontext.

Brush: Bezeichnet den Pinsel, der für das Graufärben verwendet werden soll.

OutputFunc: Ist die Adresse der Prozedurinstanz der von der Anwendung übergebenen Funktion, die den String zeichnen soll. Wenn die Funktion *TextOut* verwendet wird, um den

String darzustellen, ist *OutputFunc* ein NULL-Zeiger. Details finden sich im folgenden »Hinweis«.

Data: Bezeichnet einen Zeiger auf Daten, die an die Ausgabefunktion übergeben werden sollen. Wenn der Parameter *OutputFunc* 0 ist, muß *Data* ein Zeiger auf den String sein, der ausgegeben werden soll.

Count: Bestimmt die Anzahl der auszugebenden Zeichen. Wenn der Parameter *Count* 0 ist, berechnet *GrayString* die Länge des Strings (vorausgesetzt, daß *Data* ein Zeiger auf den String ist). Ist *Count* -1 und gibt die Funktion, auf die *OutputFunc* zeigt, 0 zurück, wird das gezeigte Bild nicht grau dargestellt.

X: Bestimmt die x-Koordinate der virtuellen Startposition des Rechtecks, das den String einschließt.

Y: Bestimmt die y-Koordinate der Startposition des Rechtecks, das den String einschließt.

Width: Bestimmt die Breite des Rechtecks, das den String einschließt, in virtuellen Einheiten. Ist der Parameter *Width* 0, berechnet *GrayString* die Breite des Bereichs, vorausgesetzt, daß *Data* ein Zeiger auf den String ist.

Height: Bestimmt die Höhe des Rechtecks, das den String einschließt, in virtuellen Einheiten. Wenn der Parameter *Height* 0 ist, berechnet *GrayString* die Höhe des Bereichs, vorausgesetzt, daß *Data* ein Zeiger auf den String ist.

Rückgabewert

Der Rückgabewert bezeichnet das Ergebnis der Funktion. Er ist ungleich Null, wenn der String dargestellt wurde. Null bedeutet, daß entweder die Funktion *TextOut* oder die von der Anwendung definierte Funktion 0 zurückgegeben hat oder daß nicht genügend Speicherplatz vorhanden war, um ein Speicherbitmap zur Darstellung des nichtaktivierbaren Zustands zu erstellen.

Hinweis

Eine Anwendung kann graugefärbte Strings auf Geräten zeichnen, die eine flächige graue Farbe unterstützen, ohne dafür die Funktion *GrayString* aufzurufen. Die Systemfarbe *COLOR_GRAYTEXT* ist die flächige graue Systemfarbe, die verwendet wird, um nichtaktivierbaren Text darzustellen.

Die Anwendung kann die Funktion *GetSysColor* aufrufen, um den Farbwert für *COLOR_GRAYTEXT* abzurufen. Wenn die Farbe von 0 (Schwarz) abweicht, kann die Anwendung die Funktion *SetTextColor* aufrufen, um die Textfarbe auf diesen Farbwert zu setzen und um dann den String direkt darzustellen. Wenn die abgerufene Farbe Schwarz ist, muß die Anwendung *GrayString* aufrufen, um den Text als nicht aktivierbar darzustellen.

Callback-Funktion

```
BOOL FAR PASCAL OutputFunc(HDC DC; DWORD Data,
int Count);
```

OutputFunc: Ein Platzhalter für den anwendungsunterstützten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

DC: Bezeichnet einen Speichergerätekontext mit einem Bitmap mit mindestens der Breite und der Höhe, die von den Parametern *Width* beziehungsweise *Height* vorgegeben wird.

Data: Zeigt auf den String, der gezeichnet werden soll.

Count: Bestimmt die Anzahl der auszugebenden Zeichen.

Rückgabewert Bei erfolgreicher Ausführung der Funktion wird ein Wert ungleich 0 zurückgeliefert, andernfalls 0.

Hinweis Diese Ausgabefunktion (*OutputFunc*) muß ein Bild relativ zu den Koordinaten (0,0) statt zu (X,Y) zeichnen. Die als Parameter *OutputFunc* übergebene Adresse muß mit der Funktion *MakeProcInstance* erzeugt worden sein.

Der Abbildungsmodus *MM_TEXT* muß ausgewählt sein, bevor diese Funktion benutzt wird.

HIBYTE

Syntax

```
BYTE HIBYTE(int Integer);
```

Beschreibung Dieses Makro liefert das höherwertige Byte des Integer-Parameters Integer.

Integer: Bestimmt den Wert, der konvertiert werden soll.

Rückgabewert Der Rückgabewert ist das höherwertige Byte des vorgegebenen Integer.

HideCaret

Syntax

```
void HideCaret(HWND Wnd);
```

Beschreibung Diese Funktion blendet das Caret aus, indem sie es vom Bildschirm löscht. Obwohl das Caret nicht mehr sichtbar ist, kann es mit der Funktion *ShowCaret* wieder dargestellt werden. Durch das Ausblenden des Caret wird seine aktuelle Form nicht beeinflußt.

HideCaret blendet das Caret nur dann aus, wenn es dem bezeichneten Fenster zugeordnet ist. Ist der Parameter *Wnd* 0, wird das Caret nur ausgeblendet, wenn ein Fenster der aktuellen Task das Caret besitzt. Das Ausblenden geschieht kumulativ. Wurde *HideCaret* fünfmal hintereinander aufgerufen wur-

de, so muß *ShowCaret* fünfmal aufgerufen werden, damit das Caret wieder dargestellt wird.

Wnd: Bezeichnet das Fenster, welches das Caret besitzt oder ist 0, um indirekt auf das Fenster in der aktuellen Task, welches das Caret besitzt, zu verweisen.

Rückgabewert Keiner.

Querverweis *ShowCaret*

HiliteMenuItem

Syntax
```
BOOL HiliteMenuItem(HWND Wnd, HMENU Menu,
WORD IDHilite, WORD Hilite);
```

Beschreibung Diese Funktion hebt eine Hauptmenüoption (Menüleiste) hervor oder setzt sie auf den Normalzustand zurück.

Wnd: Bezeichnet das Fenster, welches das Menü enthält.

Menu: Bezeichnet das Hauptmenü, welches die hervorzuhebende Option enthält.

IDHilite: Bestimmt den Integer-Bezeichner der Menüoption oder deren Offset im Menü in Abhängigkeit vom Parameter *Hilite*.

Hilite: Bestimmt, ob die Menüoption hervorgehoben oder die Hervorhebung zurückgenommen werden soll.

Möglich ist eine Kombination von *MF_HILITE* oder *MF_UNHILITE* mit *MF_BYCOMMAND* oder *MF_BYPOSITION*. Diese Werte können mit Hilfe des bitweisen ODER-Operators (|) kombiniert werden.

Sie haben die in folgender Tabelle beschriebene Bedeutung:

Wert	Bedeutung
MF_BYCOMMAND	Interpretiert *IDHilite* als Bezeichner der Menüoption (Standardinterpretation).
MF_BYPOSITION	Interpretiert *IDHilite* als Offset.
MF_HILITE	Stellt die Option hervorgehoben dar. Wenn dieser Wert nicht angegeben ist, wird die Option in den normalen Zustand zurückgesetzt.
MF_UNHILITE	Setzt die Option in den normalen Zustand zurück.

Rückgabewert Gibt an, ob die Menüoption als Ergebnis der Funktion hervorgehoben wird oder nicht. Er ist ungleich Null, wenn das Element hervorgehoben wurde, andernfalls ist er Null.

Hinweis Die Flags *MF_HILITE* und *MF_UNHILITE* können nur mit der Funktion *HiliteMenuItem*, nicht aber mit der Funktion *ModifyMenu* verwendet werden.

HIWORD

Syntax
```
WORD HIWORD(DWORD AnInteger);
```

Beschreibung Dieses Makro liefert das höherwertige WORD des 32-Bit-Integer, der vom Parameter *AnInteger* bezeichnet wird.

AnInteger: Bestimmt den zu bearbeitenden Wert.

Rückgabewert Der Rückgabewert ist das höherwertige WORD des gegebenen 32-Bit-Integer.

InflateRect

Syntax
```
void InflateRect(LPRECT Rect, int X, int Y);
```

Beschreibung Diese Funktion vergrößert oder verkleinert Höhe und Breite des angegebenen Rechtecks. Sie fügt *X* Einheiten an die linke und die rechte Seite des Rechtecks sowie *Y* Einheiten an die Ober- und Unterseite an.

Die Parameter *X* und *Y* sind vorzeichenbehaftete Werte; positive Werte vergrößern Breite und Höhe, negative Werte verringern sie.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die verändert werden soll.

X: Bestimmt den Betrag, um den die Breite des Rechtecks vergrößert oder verkleinert werden soll. Ist *X* negativ, wird die Breite verringert.

Y: Bestimmt den Betrag, um den die Höhe des Rechtecks vergrößert oder verkleinert werden soll. Ist *Y* negativ, wird die Höhe verringert.

Rückgabewert Keiner.

Hinweis Die Koordinatenwerte eines Rechtecks dürfen nicht größer als 32767 und nicht kleiner als -32767 Einheiten sein. Die Parameter *X* und *Y* müssen so gewählt werden, daß keine ungültigen Rechtecke auftreten.

InitAtomTable

Syntax
```
BOOL InitAtomTable(int Size);
```

Beschreibung Diese Funktion initialisiert eine Atom-Hash-Tabelle in der durch den Parameter *Size* vorgegebenen Größe. Wird diese

Funktion nicht aufgerufen, wird die Größe der Atom-Hash-Tabelle standardmäßig auf 37 gesetzt.

Diese Funktion sollte, wenn sie benötigt wird, vor jeder anderen Atom-Verwaltungsfunktion aufgerufen werden.

Size: Bestimmt die Größe der Atom-Hash-Tabelle in Tabelleneinträgen. Dieser Wert sollte eine Primzahl sein.

Rückgabewert Nach erfolgreicher Ausführung der Funktion wird ungleich Null zurückgeliefert, andernfalls Null.

Hinweis Benutzt eine Anwendung eine große Zahl von Atomen, so kann sie die erforderliche Zeit für das Hinzufügen eines Atoms zur Atomtabelle oder das Finden eines Atoms in der Tabelle verringern, indem sie die Tabelle vergrößert. Dies erhöht jedoch den Speicherplatz, der zur Verwaltung der Tabelle erforderlich ist.

Die Standardgröße von 37 Einträgen für die globale Atomtabelle kann nicht verändert werden.

InSendMessage

Syntax
```
BOOL InSendMessage(void);
```

Beschreibung Diese Funktion bestimmt, ob die aktuelle Fensterfunktion eine Botschaft bearbeitet, die ihr durch einen Aufruf der Funktion *SendMessage* übergeben wurde.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Fensterfunktion eine Botschaft bearbeitet, die ihr durch *SendMessage* übergeben wurde, andernfalls ist er Null.

Hinweis Anwendungen benutzen die Funktion *InSendMessage*, um festzulegen, wie auf Fehler reagiert werden soll, die bei der Verarbeitung von Botschaften durch ein inaktives Fenster auftreten.

Benutzt beispielsweise das aktive Fenster *SendMessage*, um eine Informationsanforderung an ein anderes Fenster zu senden, so kann das andere Fenster solange nicht aktiv werden, bis es die Steuerung von *SendMessage* zurückerhält. Ein inaktives Fenster kann einen Anwender nur dann von einem Fehler benachrichtigen, wenn es ein Meldungsfenster darstellt.

InsertMenu

Syntax
```
BOOL InsertMenu(HMENU Menu, WORD Position, WORD Flags,
WORD IDNewItem, LPSTR NewItem);
```

Beschreibung Diese Funktion fügt an der vom Parameter *Position* vorgegebenen Position eine neue Menüoption ein. Die anderen Optionen werden nach unten verschoben. Die Anwendung kann

den Status der Menüoption bestimmen, indem sie die Werte im Parameter *Flags* übergibt.

Menu: Bezeichnet das Menü, welches verändert werden soll.

Position: Bestimmt die Menüoption, vor der die neue Menüoption eingefügt werden soll. Die Interpretation des Parameters *Position* hängt von der Einstellung des Parameters *Flags* ab.

Flags	Position:
MF_BYPOSITION	Bestimmt die Position der existierenden Menüoption. Die erste Option im Menü steht an Position 0. Wenn *Position* -1 ist, wird die neue Menüoption am Ende des Menüs angefügt.
MF_BYCOMMAND	Bestimmt den Befehlsbezeichner der existierenden Menüoption.

Flags: Bestimmt, wie der Parameter *Position* interpretiert wird und informiert über den Status der neuen Menüoption, wenn diese in das Menü eingefügt wird. Er besteht aus einem oder mehreren der Werte, die im folgenden »Hinweis« beschrieben sind.

IDNewItem: Bestimmt entweder den Befehlsbezeichner der neuen Menüoption oder, falls *Flags* auf *MF_POPUP* gesetzt ist, das Menü-Handle des Pop-up-Menüs.

NewItem: Bestimmt den Inhalt der neuen Menüoption. Wenn *Flags* auf *MF_STRING* gesetzt ist (Voreinstellung), ist *NewItem* ein Zeiger auf einen nullterminierten String. Ist Flags stattdessen auf *MF_BITMAP* gesetzt, enthält *NewItem* ein Bitmap-Handle (*HBITMAP*) in seinem niederwertigen WORD. Wenn *Flags* auf *MF_OWNERDRAW* gesetzt ist, bezeichnet *NewItem* einen von der Anwendung gelieferten 32-Bit-Wert, den diese verwenden kann, um weitere der Menüoption zugeordnete Daten zu verwalten. Dieser 32-Bit-Wert steht der Anwendung im Feld *itemData* der Datenstruktur zur Verfügung, auf die der Parameter *lParam* der Botschaften *WM_MEASUREITEM* und *WM_DRAWITEM* zeigt. Diese Botschaften werden gesendet, wenn die Menüoption erstmals dargestellt oder wenn sie verändert wird.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Bei jeder Veränderung eines Menüs (unabhängig davon, ob es sich in einem dargestellten Fenster befindet oder nicht), sollte eine Anwendung *DrawMenuBar* aufrufen.

Jede der folgenden Gruppen enthält Flags, die nicht gleichzeitig verwendet werden können:

MF_BYCOMMAND und *MF_BYPOSITION*
MF_DISABLED, *MF_ENABLED* und *MF_GRAYED*
MF_BITMAP, *MF_STRING*, *MF_OWNERDRAW* und *MF_SEPARATOR*
MF_MENUBARBREAK und *MF_MENUBREAK*
MF_CHECKED und *MF_UNCHECKED*

Folgende Flags können im Parameter *Flags* gesetzt werden:

Flag	Bedeutung
MF_BITMAP	Verwendet ein Bitmap als Menüoption. Das niederwertige WORD des Parameters *NewItem* enthält das Handle des Bitmaps.
MF_BYCOMMAND	Bestimmt, daß der Parameter *Position* die Nummer des Steuerelementbezeichners der Menüoption angibt (Voreinstellung).
MF_BYPOSITION	Bestimmt, daß der Parameter *Position* statt einer ID-Nummer eine zu verändernde Menüoption bezeichnet.
MF_CHECKED	Plaziert eine Auswahlmarkierung neben der Menüoption. Stellt die Anwendung Auswahlmarkierung-Bitmaps zur Verfügung (siehe auch *SetMenuItemBitmaps*), bewirkt das Setzen dieses Flags die Darstellung des Bitmaps neben der Menüoption.
MF_DISABLED	Sperrt die Menüoption, so daß sie nicht ausgewählt werden kann, versetzt sie jedoch nicht in den nichtaktivierbaren Zustand (schreibt sie nicht grau).
MF_ENABLED	Aktiviert die Menüoption, so daß sie ausgewählt werden kann und hebt die Nichtaktivierbarkeit auf.
MF_GRAYED	Sperrt die Menüoption, so daß sie nicht ausgewählt werden kann, und versetzt sie in den nichtaktivierbaren Zustand.
MF_MENUBARBREAK	Ist gleichbedeutend mit *MF_MENUBREAK* mit der Ausnahme, daß bei Pop-up-Menüs die neue Spalte von der alten Spalte durch eine vertikale Linie getrennt wird.
MF_MENUBREAK	Plaziert die Menüoption für statische Menüleistenelemente in einer neuen Zeile. Bei Pop-up-Menüs wird die Menüoption in einer neuen Spalte ohne Spaltentrennlinie plaziert.

Flag	Bedeutung
MF_OWNERDRAW	Bestimmt, daß die Option eine besitzergezeichnete dargestellte Menüoption ist. Das Fenster, dem das Menü gehört, empfängt eine Botschaft *WM_MEASUREITEM*, wenn das Menü zum ersten Mal dargestellt wird, um Höhe und Breite der Menüoption abzurufen. Die Botschaft *WM_DRAWITEM* wird immer an die Anwendung gesendet, wenn diese das Erscheinungsbild der Menüoption aktualisieren muß. Diese Option ist für Hauptmenüoptionen nicht gültig.
MF_POPUP	Bestimmt, daß der Menüoption ein Pop-up-Menü zugeordnet ist. Der Parameter *IDNewItem* bezeichnet ein Handle auf das Pop-up-Menü, welches der Option zugeordnet werden soll. Das Flag *MF_OWNERDRAW* sollte verwendet werden, um einer Pop-up-Menüoption entweder ein Haupt-Pop-up-Menü oder ein hierarchisches Pop-up- Menü zuzuordnen.
MF_SEPARATOR	Zeichnet eine horizontale Trennlinie. Man kann dieses Flag in einem Pop-up- Menü benutzen. Diese Linie kann nicht gesperrt, hervorgehoben oder in nichtaktivierbaren Zustand versetzt werden. Windows ignoriert die Parameter *NewItem* und *IDNewItem*.
MF_STRING	Bestimmt, daß die Menüoption ein String ist. Der Parameter *NewItem* zeigt auf den String für die Menüoption.
MF_UNCHECKED	Plaziert eine Nichtauswahlmarkierung neben der Option (Voreinstellung). Stellt die Anwendung entsprechende Bitmaps zur Verfügung (siehe auch *SetMenuItemBitmaps*), bewirkt das Setzen dieses Flags die Darstellung des Bitmaps neben der Menüoption.

Querverweis *DrawMenuBar, SetMenuItemBitmaps, WM_DRAWITEM, WM_-MEASUREITEM*

IntersectClipRect

Syntax
```
int IntersectClipRect(HDC DC, int X1, int Y1, int X2,
int Y2);
```

Beschreibung Diese Funktion erzeugt eine neue Clipping-Region, indem sie den Überschneidungsbereich der aktuellen Region und des

von *X1*, *Y1*, *X2* und *Y2* gegebenen Rechtecks bildet. Windows formatiert alle folgenden Ausgaben so, daß sie in die neuen Grenzen passen.

DC: Bezeichnet den neuen Gerätekontext.

X1, Y1: Bestimmt die virtuelle x- bzw. y-Koordinate der oberen linken Ecke des Rechtecks.

X2, Y2: Bestimmt die virtuelle x- bzw. y-Koordinate der unteren rechten Ecke des Rechtecks.

Rückgabewert Der Rückgabewert gibt den Typ der neuen Clipping-Region an. Er kann einen der folgenden Werte annehmen:

COMPLEXREGION	Neue Clipping-Region hat überlappende Grenzen.
ERROR	Gerätekontext ist ungültig.
NULLREGION	Neue Clipping-Region ist leer.
SIMPLEREGION	Neue Clipping-Region hat keine überlappenden Grenzen.

Hinweis Die Breite des Rechtecks, gebildet durch die Differenz X2 - X1, darf 32767 Einheiten nicht überschreiten. Dieser Höchstwert gilt auch für die Höhe des Rechtecks.

IntersectRect

Syntax
```
int IntersectRect(LPRECT DestRect, LPRECT Src1Rect,
LPRECT Src2Rect);
```

Beschreibung Diese Funktion erzeugt den Überschneidungsbereich von zwei existierenden Rechtecken. Der Überschneidungsbereich ist das größte Rechteck, welches in beiden Rechtecken enthalten ist.

IntersectRect kopiert die Koordinaten des neuen Rechtecks in die *RECT*-Datenstruktur, auf die *DestRect* zeigt.

DestRect: Zeigt auf die *RECT*-Datenstruktur, welche den Überschneidungsbereich aufnehmen soll.

Src1Rect, Src2Rect: Zeigen auf *RECT*-Datenstrukturen, welche die Quellrechtecke enthalten.

Rückgabewert Der Rückgabewert gibt den Überschneidungsbereich von zwei Rechtecken an. Er ist ungleich 0, wenn der Überschneidungsbereich der beiden Rechtecke nicht leer ist, andernfalls 0.

InvalidateRect

Syntax
```
void InvalidateRect(HWND Wnd, LPRECT Rect, BOOL Erase);
```

Beschreibung Diese Funktion macht den Client-Bereich innerhalb des gegebenen Rechtecks ungültig, indem dieses der Aktualisierungs-

region von Windows hinzugefügt wird. Das ungültig gemachte Rechteck wird genauso wie alle anderen Bereiche in der Aktualisierungsregion zum Zeichnen bereitgestellt, wenn die nächste Botschaft *WM_PAINT* eintrifft. Die ungültig werdenden Bereiche sammeln sich in der Aktualisierungsregion solange an, bis die Region beim Eintreffen der nächsten Botschaft *WM_PAINT* bearbeitet oder sie mit *ValidateRect* oder *ValidateRgn* auf ihre Gültigkeit überprüft wird.

Der Parameter *Erase* gibt an, ob der Hintergrund innerhalb des Aktualisierungsbereiches gelöscht werden soll, wenn die Aktualisierungsregion bearbeitet wird. Ist *Erase* ungleich Null, wird der Hintergrund gelöscht, sobald die Funktion *BeginPaint* aufgerufen wird, ist der Parameter Null, bleibt der Hintergrund unverändert. Ist *Erase* für einen Teil der Aktualisierungsregion ungleich Null, wird der Hintergrund der gesamten Region gelöscht, nicht nur der des bezeichneten Bereichs.

Wnd: Bezeichnet das Fenster, dessen Aktualisierungsregion verändert werden soll.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die das Rechteck in Client-Koordinaten enthält, das an die Aktualisierungsregion angefügt werden soll. Ist der Parameter *Rect* NULL, wird der gesamte Client-Bereich der Aktualisierungsregion hinzugefügt.

Erase: Bestimmt, ob der Hintergrund innerhalb der Aktualisierungsregion gelöscht werden soll.

Rückgabewert Keiner.

Hinweis Windows sendet immer dann eine Botschaft *WM_PAINT* an ein Fenster, wenn dessen Aktualisierungsregion nicht leer ist und in der Anwendungs-Warteschlange keine weiteren Botschaften für dieses Fenster vorliegen.

Querverweis *ValidateRect, ValidateRgn, WM_PAINT*

InvalidateRgn

Syntax
```
void InvalidateRgn(HWND Wnd, HRGN Rgn, BOOL Erase);
```

Beschreibung Diese Funktion macht den Client-Bereich innerhalb der angegebenen Region ungültig, indem sie diese der Aktualisierungsregion des gegebenen Fensters hinzufügt. Die ungültig gemachte Region wird genau wie alle anderen Bereiche in der Aktualisierungsregion zum Zeichnen bereitgestellt, wenn die nächste Botschaft *WM_PAINT* eintrifft. Die ungültig werdenden Regionen sammeln sich solange in der Aktualisierungsregion an, bis die Region beim nächsten Eintreffen einer Botschaft *WM_PAINT* bearbeitet oder sie mit *ValidateRect* oder *ValidateRgn* auf ihre Gültigkeit überprüft wird.

Der Parameter *Erase* bestimmt, ob der Hintergrund innerhalb der Aktualisierungsregion gelöscht werden soll, wenn die Aktualisierungsregion bearbeitet wird. Ist *Erase* ungleich Null, wird der Hintergrund beim Aufruf der Funktion *BeginPaint* gelöscht, ist *Erase* Null, bleibt der Hintergrund unverändert. Wenn *Erase* für irgendeinen Teil der Aktualisierungsregion ungleich Null ist, wird der Hintergrund in der gesamten Region gelöscht, nicht nur der des bezeichneten Bereichs.

Wnd: Bezeichnet das Fenster, dessen Aktualisierungsregion verändert werden soll.

Rgn: Bezeichnet die Region, die an die Aktualisierungsregion angefügt werden soll. Es wird davon ausgegangen, daß die Region in Client-Koordinaten angegeben ist.

Erase: Bestimmt, ob der Hintergrund innerhalb der Aktualisierungsregion gelöscht werden soll.

Rückgabewert Keiner.

Hinweis Windows sendet immer dann eine Botschaft *WM_PAINT* an ein Fenster, wenn seine Aktualisierungsregion nicht leer ist und in der Anwendungswarteschlange keine anderen Botschaften für das Fenster vorliegen. Die angegebene Region muß vorher mit einer entsprechenden Funktion erzeugt worden sein.

Querverweis *ValidateRect, ValidateRgn, WM_PAINT*

InvertRect

Syntax
```
void InvertRect(HDC DC, LPRECT Rect);
```

Beschreibung Diese Funktion invertiert die Inhalte des gegebenen Rechtecks. Auf Monochrombildschirmen färbt sie weiße Pixel schwarz und schwarze Pixel weiß. Auf Farbbildschirmen hängt die Inversion davon ab, wie Farben für den Bildschirm erzeugt werden. Der zweimalige Aufruf von *InvertRect* für dasselbe Rechteck stellt den Bildschirm mit den vorherigen Farben wieder her.

DC: Bezeichnet den Gerätekontext.

Rect: Zeigt auf eine *RECT*-Datenstruktur, welche die virtuellen Koordinaten des zu invertierenden Rechtecks enthält.

Rückgabewert Keiner.

Hinweis Die Funktion vergleicht die Werte der Felder *top*, *bottom*, *left* und *right* des bezeichneten Rechtecks. Wenn *bottom* kleiner oder gleich *top* ist oder *right* kleiner oder gleich *left* ist, wird das Rechteck nicht invertiert.

InvertRgn

Syntax
```
BOOL InvertRgn(HDC DC, HRGN Rgn);
```

Beschreibung Diese Funktion invertiert die Farben in der Region, die der Parameter *Rgn* bezeichnet.

Auf Monochrombildschirmen färbt die Funktion *InvertRgn* weiße Pixel schwarz und schwarze Pixel weiß. Auf Farbbildschirmen hängt die Inversion davon ab, wie Farben für den Bildschirm erzeugt werden.

DC: Bezeichnet den Gerätekontext.

Rgn: Bezeichnet die Region, die invertiert werden soll. Die Koordinaten für die Region sind in gerätespezifischen Einheiten angegeben.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

IsCharAlpha

Syntax
```
BOOL IsCharAlpha(char AChar);
```

Beschreibung Diese Funktion stellt fest, ob ein Zeichen ein alphabetisches Zeichen ist.

Diese Feststellung wird vom Sprachtreiber getroffen, der vom Anwender im Setup oder in der Systemsteuerung ausgewählt wurde. Auch die deutschen Umlaute sowie das ß sind alphabetisch.

AChar: Bestimmt das Zeichen, das überprüft werden soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Zeichen alphabetisch ist, andernfalls ist er Null.

IsCharAlphaNumeric

Syntax
```
BOOL IsCharAlphaNumeric(char AChar);
```

Beschreibung Diese Funktion stellt fest, ob ein Zeichen ein alphanumerisches (alphabetisch oder eines der dezimalen Ziffern) Zeichen ist.

Diese Feststellung wird vom Sprachtreiber getroffen, der vom Anwender im Setup oder in der Systemsteuerung ausgewählt wurde.

AChar: Das Zeichen, das überprüft werden soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Zeichen alphanumerisch ist, andernfalls ist er Null.

IsCharLower

Syntax
```
BOOL IsCharLower(char AChar);
```

Beschreibung Diese Funktion stellt fest, ob das Zeichen in Kleinschreibung vorliegt.

Diese Feststellung wird vom Sprachtreiber getroffen, der vom Anwender im Setup oder in der Systemsteuerung ausgewählt wurde.

AChar: Das Zeichen, das überprüft werden soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Zeichen in Kleinschreibung vorliegt, andernfalls ist er Null.

IsCharUpper

Syntax
```
BOOL IsCharUpper(char AChar);
```

Beschreibung Diese Funktion stellt fest, ob das Zeichen in Großschreibung vorliegt.

Diese Feststellung wird vom Sprachtreiber getroffen, der vom Anwender im Setup oder in der Systemsteuerung ausgewählt wurde.

AChar: Das Zeichen, das überprüft werden soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Zeichen in Großschreibung vorliegt, andernfalls ist er Null.

IsChild

Syntax
```
BOOL IsChild(HWND Parent, HWND Wnd);
```

Beschreibung Diese Funktion ermittelt, ob das vom Parameter *Wnd* angegebene Fenster ein untergeordnetes Fenster oder ein anderer direkter Abkömmling des Fensters ist, welches der Parameter *Parent* bezeichnet.

Ein untergeordnetes Fenster ist der direkte Abkömmling des angegebenen übergeordneten Fensters, wenn dieses übergeordnete Fenster in der Kette der übergeordneten Fenster steht, die vom ursprünglichen Pop-up-Fenster zum untergeordneten Fenster führt.

Parent: Bezeichnet ein Fenster.

Wnd: Bezeichnet das Fenster, das überprüft werden soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das vom Parameter *Wnd* bezeichnete Fenster ein untergeordnetes Fenster des vom Parameter *Parent* bezeichneten Fensters ist, andernfalls ist er Null.

IsClipboardFormatAvailable

Syntax
```
BOOL IsClipboardFormatAvailable(WORD Format);
```

Beschreibung Diese Funktion ermittelt, ob Daten eines bestimmten Typs in der Zwischenablage existieren.

Format: Bestimmt ein registriertes Zwischenablageformat. Informationen über Zwischenablageformate finden sich im Abschnitt »CF_XXXX« in Kapitel 4 »Windows-Stile und -Konstanten«.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn Daten im bezeichneten Format vorliegen, andernfalls ist er Null.

Hinweis Diese Funktion wird normalerweise während der Verarbeitung der Botschaften *WM_INITMENU* oder *WM_INITMENUPOPUP* verwendet um festzustellen, ob die Zwischenablage Daten enthält, die in die Anwendung eingefügt werden können. Sind solche Daten vorhanden, versetzt die Anwendung üblicherweise den Befehl *Paste* (Kopieren) in ihrem Edit-Menü (Bearbeiten) in den aktivierbaren Zustand.

Querverweis *SetClipboardData*

IsDialogMessage

Syntax
```
BOOL IsDialogMessage(HWND Dlg, LPMSG Msg);
```

Beschreibung Diese Funktion ermittelt, ob die angegebene Botschaft für das durch *Dlg* bezeichnete nicht-modale Dialogfenster bestimmt ist und verarbeitet die Botschaft in diesem Fall automatisch.

Wenn *IsDialogMessage* eine Botschaft verarbeitet, prüft sie auf Tastaturbotschaften und konvertiert sie in Auswahlbefehle für das zugehörige Dialogfenster. Z.B. wählt die Tabulatortaste das nächste Steuerelement oder die nächste Gruppe von Steuerelementen aus. Die Pfeiltaste nach unten wählt das nächste Steuerelement in einer Gruppe aus. Wenn eine Botschaft von *IsDialogMessage* verarbeitet wurde, darf sie nicht an die Funktionen *TranslateMessage* oder *DispatchMessage* übergeben werden, da *IsDialogMessage* die gesamte notwendige Übersetzung und Verteilung vornimmt. *IsDialogMessage* sendet eine Botschaft *WM_GETDLGCODE* an die Dialogfunktion, um zu bestimmen, welche Tasten verarbeitet werden sollen.

Dlg: Bezeichnet das Dialogfenster.

Msg: Zeigt auf eine *MSG*-Datenstruktur, die die zu überprüfende Botschaft enthält.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Botschaft verarbeitet worden ist, andernfalls ist er Null.

Hinweis Obwohl *IsDialogMessage* für nicht-modale Dialogfenster vorgesehen ist, kann diese Funktion für jedes Fenster benutzt werden, das Steuerelemente enthält, um dieselbe Tastaturauswahl wie in einem Dialogfenster zu ermöglichen.

IsDlgButtonChecked

Syntax
```
WORD IsDlgButtonChecked(HWND Dlg, int IDButton);
```

Beschreibung Diese Funktion stellt fest, ob ein Schalter eine Auswahlmarkierung hat und ob ein dreistufiger Schalter nichtaktivierbar, ausgewählt oder keines von beidem ist.

Die Funktion *IsDlgButtonChecked* übergibt eine Botschaft *BM_GETCHECK* an den Schalter.

Dlg: Bezeichnet das Dialogfenster, das den Schalter enthält.

IDButton: Bestimmt den Integer-Bezeichner des Schalters.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn der angegebene Schalter markiert ist, andernfalls ist er 0.

Bei dreistufigen Schaltern ist der Rückgabewert 2, wenn der Schalter nichtaktivierbar ist, er ist 1, wenn der Schalter markiert ist und 0, wenn er nicht markiert ist.

IsIconic

Syntax
```
BOOL IsIconic(HWND Wnd);
```

Beschreibung Diese Funktion ermittelt, ob ein Fenster zum Symbol verkleinert ist.

Wnd: Bezeichnet das Fenster.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Fenster verkleinert ist, andernfalls Null.

IsRectEmpty

Syntax
```
BOOL IsRectEmpty(LPRECT Rect);
```

Beschreibung Diese Funktion stellt fest, ob das bezeichnete Rechteck leer ist oder nicht. Ein Rechteck ist leer, wenn Höhe und/oder Breite 0 sind.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die das angegebene Rechteck enthält.

Rückgabewert Der Rückgabewert ist ungleich *Null*, wenn das Rechteck leer ist, andernfalls *Null*.

IsWindow

Syntax `BOOL IsWindow(HWND Wnd);`

Beschreibung Diese Funktion stellt fest, ob das Fenster, welches durch den Parameter *Wnd* bezeichnet ist, ein gültiges existierendes Fenster ist.

Wnd: Bezeichnet das Fenster.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn *Wnd* ein gültiges Fenster ist, andernfalls 0.

IsWindowEnabled

Syntax `BOOL IsWindowEnabled(HWND Wnd);`

Beschreibung Diese Funktion ermittelt, ob das durch *Wnd* bezeichnete Fenster zur Entgegennahme von Maus- und Tastatureingaben befähigt ist.

Wnd: Bezeichnet das Fenster.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Fenster aktiviert ist, andernfalls Null.

Hinweis Ein untergeordnetes Fenster empfängt nur Eingaben, wenn es sowohl aktiviert als auch sichtbar ist.

IsWindowVisible

Syntax `BOOL IsWindowVisible(HWND Wnd);`

Beschreibung Diese Funktion gibt einen Wert ungleich Null zurück, wenn ein Fenster mit der Funktion *ShowWindow* sichtbar gemacht wurde. Auch wenn das bezeichnete Fenster vollständig von einem anderen untergeordneten oder Pop-up-Fenster verdeckt ist, ist der Rückgabewert ungleich Null.

Wnd: Bezeichnet das Fenster.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das gegebene Fenster auf dem Bildschirm vorhanden ist, andernfalls ist er Null.

IsZoomed

Syntax `BOOL IsZoomed(HWND Wnd);`

Beschreibung Diese Funktion stellt fest, ob ein Fenster auf die volle Größe gesetzt ist oder nicht.

Wnd: Bezeichnet das Fenster.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Fenster vergrössert ist, andernfalls Null.

KillTimer

Syntax
```
BOOl KillTimer(HWND Wnd, int IDEvent);
```

Beschreibung Diese Funktion eliminiert das Timer-Ereignis, das durch die Parameter *Wnd* und *IDEvent* bezeichnet wird. Alle anstehenden *WM_TIMER*- Botschaften, die mit dem Timer verbunden sind, werden aus der Botschaftswarteschlange entfernt.

Wnd: Bezeichnet das Fenster, das mit dem angegebenen Timer-Ereignis verbunden ist. Dies muß derselbe Wert sein, der beim Aufruf der Funktion *SetTimer* als Parameter *Wnd* übergeben wurde, wenn diese das Timer-Ereignis erzeugt hat.

IDEvent: Bestimmt das Timer-Ereignis, das eliminiert werden soll. Hat die Anwendung *SetTimer* mit auf 0 gesetztem Parameter *Wnd* aufgerufen, muß es der der von *SetTimer* zurückgegeben Ereignisbezeichner sein. War der Parameter *Wnd* von *SetTimer* ein gültiges Fenster-Handle, muß *IDEvent* der Wert des Parameters *IDEvent* sein, der an *SetTimer* übergeben wurde.

Return Value Der Rückgabewert ist ungleich Null, wenn das Ereignis eliminiert wurde und Null, wenn die Funktion das angegebene Timer-Ereignis nicht finden konnte.

Querverweis *SetTimer*

_lclose

Syntax
```
int _lclose(int FileHandle);
```

Beschreibung Diese Funktion schließt die durch den Parameter *FileHandle* bezeichnete Datei. Die Datei ist dann nicht länger für Lese- oder Schreiboperationen verfügbar. Das Argument *FileHandle* wird durch den Funktionsaufruf zurückgeliefert, der diese Datei erzeugt oder zuletzt geöffnet hat.

FileHandle: Bestimmt das DOS-Datei-Handle der Datei, die geschlossen werden soll.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion die Datei geschlossen hat, im Fall eines Fehlers ist der Wert -1.

_lcreat

Syntax
```
int _lcreat(LPSTR PathName, int Attribute);
```

Beschreibung Diese Funktion öffnet eine Datei mit dem durch den Parameter *PathName* bezeichneten Namen. Der Parameter *Attribute*

bestimmt das Attribut der Datei. Wenn keine Datei mit dem angegebenen Namen existiert, erzeugt die Funktion eine neue Datei und öffnet sie für Schreiboperationen. Beim Öffnen der Datei durch *_lcreat* wird der Dateizeiger auf den Anfang der Datei gesetzt.

PathName: Zeigt auf einen nullterminierten String, der den Namen der zu öffnenden Datei enthält. Der String muß aus Zeichen des ANSI-Zeichensatzes bestehen.

Attribute: Bestimmt das Dateiattribut. Der Parameter muß einen der folgenden Werte annehmen:

Wert	Bedeutung
0	Kann ohne Einschränkung gelesen oder beschrieben werden.
1	Schreibgeschützt; kann nicht zum Schreiben geöffnet werden; eine Datei gleichen Namens kann nicht erzeugt werden.
2	Unsichtbar; wird bei der Verzeichnissuche nicht angezeigt.
3	System; wird bei der Verzeichnissuche nicht angezeigt.

Rückgabewert Der Rückgabewert gibt ein MS-DOS-Datei-Handle an, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls wird -1 zurückgeliefert.

LimitEmsPages

Syntax
```
void LimitEmsPages(DWORD Kbytes);
```

Beschreibung Diese Funktion begrenzt den Umfang des Expanded Memory, den Windows einer Anwendung zuweisen wird. Der Umfang des Expanded Memory, den die Anwendung durch direkten Aufruf des INT 67H ansprechen kann, wird nicht begrenzt.

Kbytes: Bestimmt die Anzahl KByte des Expanded Memory, auf die die Anwendung zugreifen kann.

Rückgabewert Keiner.

Hinweis *LimitEmsPages* wirkt nur, wenn Expanded Memory installiert ist und von Windows genutzt wird. Greift Windows nicht auf Expanded Memory zu, hat die Funktion keine Wirkung.

LineDDA

Syntax
```
void LineDDA(int X1, int Y1, int X2, int Y2,
FARPROC LineFunc, LPSTR Data);
```

Beschreibung Diese Funktion berechnet alle aufeinander folgenden Punkte einer Linie vom Anfangspunkt (*X1*, *Y1*) bis zum Endpunkt (*X2*, *Y2*). Der Endpunkt ist nicht in die Linie eingeschlossen. Für

jeden Punkt der Linie ruft die Funktion *LineDDA* die anwendungsunterstützte Funktion auf, auf die der Parameter *LineFunc* zeigt und übergibt dieser Funktion die Koordinaten des aktuellen Punktes und den Parameter *Data*.

X1, Y1: Bestimmt die virtuelle x- bzw. y-Koordinate des ersten Punktes.

X2, Y2: Bestimmt die virtuelle x- bzw. y-Koordinate des Endpunktes.

LineFunc: Adresse der Prozedurinstanz der Callback-Funktion.

Data: Zeigt auf die Daten für die Callback-Funktion.

Rückgabewert Keiner.

Hinweis Die Adresse, die mit dem Parameter *LineFunc* übergeben wird, muß vorab durch den Aufruf der Funktion *MakeProcInstance* erzeugt werden.

Callback-Funktion

```
void FAR PASCAL LineFunc(int X, int Y, LPSTR Data);
```

LineFunc: Ein Platzhalter für den Namen der anwendungsunterstützten Funktion. Der wirkliche Funktionsbezeichner muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Parameter	Definition
X	Bestimmt die X-Koordinate des aktuellen Punktes.
Y	Bestimmt die Y-Koordinate des aktuellen Punktes.
Data	Zeigt auf die Anwendungsdaten.

Rückgabewert Die Funktion kann jede gewünschte Aufgabe übernehmen. Sie hat keinen Rückgabewert.

LineTo

Syntax

```
BOOL LineTo(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion zeichnet eine Linie von der aktuellen Position bis zum Punkt, der durch die Parameter *X* und *Y* angegeben wird. Der Endpunkt ist nicht in der Linie enthalten. Die Linie wird mit dem ausgewählten Stift gezeichnet. Wenn kein Fehler auftritt, wird die aktuelle Position auf (*X*, *Y*) gesetzt.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate des Endpunkts der Linie.

Y: Bestimmt die virtuelle y-Koordinate des Endpunkts der Linie.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Linie gezeichnet wurde, andernfalls Null.

_llseek

Syntax
```
LONG _llseek(int FileHandle, LONG Offset, int Origin);
```

Beschreibung Diese Funktion positioniert den Zeiger in einer vorher geöffneten Datei neu. Der Parameter *Origin* legt die Startposition in der Datei fest und *Offset* gibt an, wie weit (in Bytes) die Funktion den Zeiger verschieben soll.

FileHandle: Bestimmt das DOS-Datei-Handle der Datei.

Offset: Bestimmt die Anzahl Bytes, um die der Zeiger verschoben werden soll.

Origin: Bestimmt Startposition und Richtung der Verschiebung. Der Parameter muß einen der in folgender Tabelle aufgeführten Werte annehmen:

Wert	Bedeutung
0	Bewegt den Zeiger *Offset* Bytes, ausgehend vom Anfang der Datei.
1	Bewegt den Zeiger *Offset* Bytes, ausgehend von der aktuellen Position in der Datei.
2	Bewegt den Zeiger rückwärts *Offset* Bytes, ausgehend vom Ende der Datei.

Rückgabewert Bei erfolgreicher Ausführung wird der neue Offset des Zeigers (in Bytes), ausgehend vom Anfang der Datei, zurückgegeben, andernfalls -1.

Hinweis Beim Öffnen einer Datei wird der Dateizeiger auf den Anfang der Datei positioniert.

Die Funktion *_llseek* erlaubt direkten Zugriff auf den Inhalt einer Datei, indem der Zeiger um einen vorgegebenen beliebigen Betrag bewegt wird, ohne Daten zu lesen.

LoadAccelerators

Syntax
```
HANDLE LoadAccelerators(HANDLE Instance, LPSTR
TableName);
```

Beschreibung Diese Funktion lädt die durch den Parameter *TableName* bezeichnete Accelerator-Tabelle aus der ausführbaren Datei, die dem durch *Instance* angegebenen Modul zugeordnet ist.

Die Funktion *LoadAccelerators* lädt die Tabelle nur, wenn sie nicht schon vorher geladen wurde, andernfalls vergibt sie ein Handle zur existierenden Tabelle.

Instance: Bezeichnet eine Instanz des Moduls, dessen ausführbare Datei die Accelerator-Tabelle enthält.

TableName: Zeigt auf einen nullterminierten String, der den Namen der Accelerator-Tabelle enthält.

Rückgabewert Wenn die Funktion erfolgreich ausgeführt wurde, bezeichnet der Rückgabewert die geladene Accelerator-Tabelle, ansonsten ist er 0.

LoadBitmap

Syntax
```
HBITMAP LoadBitmap(HANDLE Instance, LPSTR BitmapName);
```

Beschreibung Diese Funktion lädt die durch *BitmapName* bezeichnete Bitmap-Ressource aus der ausführbaren Datei, die dem durch *Instance* angegebenen Modul zugeordnet ist.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei das Bitmap enthält.

BitmapName: Zeigt auf einen nullterminierten String, der den Namen des Bitmap enthält.

Rückgabewert Der Rückgabewert bezeichnet das angegebene Bitmap. Er ist 0, wenn das Bitmap nicht existiert.

Hinweis Eine Anwendung muß die Funktion *DeleteObject* aufrufen, um jedes Bitmap-Handle zu löschen, das von der Funktion *LoadBitmap* zurückgeliefert wurde. Das gleiche gilt für die vordefinierten Bitmaps, die nachfolgend beschrieben werden.

Die Funktion *LoadBitmap* kann auch verwendet werden, um auf die vordefinierten Bitmaps, die von Windows bereitgestellt werden, zuzugreifen. Der Parameter *Instance* muß hierzu auf 0 gesetzt werden und der Parameter *BitmapName* einen der folgenden Werte annehmen:

OBM_BTNCONERS	*OBM_OLD_RESTORE*
OBM_BTSIZE	*OBM_OLD_RGARROW*
OBM_CHECK	*OBM_OLD_UPARROW*
OBM_CHECKBOXES	*OBM_OLD_ZOOM*
OBM_CLOSE	*OBM_REDUCE*
OBM_COMBO	*OBM_REDUCED*
OBM_DNARROW	*OBM_RESTORE*
OBM_DNARROWD	*OBM_RESTORED*
OBM_LFARROW	*OBM_RGARROW*
OBM_LFARROWD	*OBM_RGARROWD*
OBM_MNARROW	*OBM_SIZE*
OBM_OLD_CLOSE	*OBM_UPARROW*
OBM_OLD_DNARROW	*OBM_UPARROWD*
OBM_OLD_LFARROW	*OBM_ZOOM*
OBM_OLD_REDUCE	*OBM_ZOOMD*

Bitmap-Namen, die mit *OBM_OLD* beginnen, bezeichnen Bitmaps, die von Windows-Versionen vor der Version 3.0 benutzt werden. Der Parameter *BitmapName* kann auch einen Wert vom Typ *MakeIntResource* beinhalten. Ist dies der Fall, muß die ID im niederwertigen WORD von *BitmapName* stehen und das höherwertige WORD auf 0 gesetzt werden.

LoadCursor

Syntax

```
HCURSOR LoadCursor(HANDLE Instance, LPSTR CursorName);
```

Beschreibung

Diese Funktion lädt die durch den Parameter *CursorName* bezeichnete Mauszeigerressource aus der ausführbaren Datei, die dem durch den Parameter *Instance* angegebenen Modul zugeordnet ist. Die Funktion lädt den Mauszeiger nur in den Speicher, wenn er nicht schon vorher geladen wurde, andernfalls vergibt sie ein Handle zur existierenden Ressource.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei den Mauszeiger enthält.

CursorName: Zeigt auf einen nullterminierten String, der den Namen der Mauszeigerressource enthält.

Rückgabewert

Bei erfolgreicher Ausführung bezeichnet der Rückgabewert den neugeladenen Mauszeiger, andernfalls ist er 0.

Hinweis

Die Funktion *LoadCursor* liefert nur dann ein gültiges Mauszeiger-Handle, wenn der Parameter *CursorName* eine Mauszeigerressource bezeichnet. Bezeichnet *CursorName* eine andere Art von Ressource (z.B. ein Symbol), ist der Rückgabewert nicht 0, obwohl er kein gültiges Mauszeiger-Handle ist.

LoadCursor kann ebenfalls verwendet werden, um auf einen von Windows bereitgestellten, vordefinierten Mauszeiger zuzugreifen. Dazu muß der Parameter *Instance* auf 0 gesetzt sein und der Parameter *CursorName* einen der folgenden Werte annehmen:

Wert	Bedeutung
IDC_ARROW	Standardpfeilzeiger.
IDC_CROSS	Fadenkreuzzeiger.
IDC_IBEAM	I-Balkenzeiger in Texten.
IDC_ICON	Leeres Symbol.
IDC_SIZE	Vierpfeilzeiger. Die Pfeile zeigen nach Norden, Osten, Süden und Westen.
IDC_SIZENESW	Doppelpfeilzeiger. Die Pfeile zeigen nach Nordost und Südwest.
IDC_SIZENS	Doppelpfeilzeiger. Die Pfeile zeigen nach Norden und Süden.

Wert	Bedeutung
IDC_SIZENWSE	Doppelpfeilzeiger. Die Pfeile zeigen nach Nordwest und Südost.
IDC_SIZEWE	Doppelpfeilzeiger. Die Pfeile zeigen nach Westen und Osten.
IDC_UPARROW	Vertikaler Pfeilzeiger nach oben.
IDC_WAIT	Sanduhrzeiger.

Der Parameter *CursorName* kann auch einen Wert vom Typ *MakeIntResource* enthalten. Ist dies der Fall, muß die ID im niederwertigen WORD von *CursorName* stehen und das höherwertige WORD muß auf 0 gesetzt werden.

LoadIcon

Syntax

```
HICON LoadIcon(HANDLE Instance, LPSTR IconName);
```

Beschreibung Diese Funktion lädt die durch den Parameter *IconName* bezeichnete Symbolressource aus der ausführbaren Datei, die dem durch den Parameter *Instance* bezeichneten Modul zugeordnet ist. Die Funktion lädt das Symbol nur, wenn es nicht schon vorher geladen wurde, andernfalls liefert sie ein Handle zur existierenden Ressource.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei das Symbol enthält.

IconName: Zeigt auf einen nullterminierten String, der den Namen der Symbolressource enthält.

Rückgabewert Bei erfolgreicher Ausführung bezeichnet der Rückgabewert eine Symbolressource, andernfalls ist er 0.

Hinweis Die Funktion *LoadIcon* kann auch verwendet werden, um auf die von Windows bereitgestellten vordefinierten Symbole zuzugreifen. Dazu muß der Parameter *Instance* auf 0 gesetzt werden und der Parameter *IconName* einen der folgenden Werte annehmen:

Wert	Bedeutung
IDI_APPLICATION	Voreingestelltes Anwendungssymbol.
IDI_ASTERISK	Stern (für Informationsmeldungen).
IDI_EXCLAMATION	Ausrufezeichen (für Warnmeldungen).
IDI_HAND	Symbol in Handform (für ernste Warnmeldungen).
IDI_QUESTION	Fragezeichen (für Anwenderabfragen).

Der Parameter *IconName* kann auch einen Wert vom Typ *MakeIntResource* enthalten. Ist dies der Fall, muß die ID im

niederwertigen WORD von *IconName* stehen und das höherwertige WORD auf 0 gesetzt werden.

LoadLibrary

Syntax

```
HANDLE LoadLibrary(LPSTR LibFileName);
```

Beschreibung Diese Funktion lädt das Bibliotheksmodul, das in der bezeichneten Datei enthalten ist und liefert ein Handle zur Instanz des geladenen Moduls.

LibFileName: Zeigt auf einen nullterminierten String, der den Namen der Bibliotheksdatei enthält.

Rückgabewert Der Rückgabewert bezeichnet bei erfolgreicher Ausführung die Instanz des geladenen Bibliothekmoduls, andernfalls hat er einen Wert kleiner als 32, der den Fehler bezeichnet. Folgende Fehlerwerte können zurückgeliefert werden:

Wert	Bedeutung
0	Zu wenig Speicher.
2	Datei nicht gefunden.
3	Pfad nicht gefunden.
5	Versuch, eine Task dynamisch einzubinden.
6	Bibliothek benötigt getrennte Datensegmente für jede Task.
10	Falsche Windows-Version.
11	Ungültige .EXE Datei (Keine Windows .EXE oder Fehler in der .EXE).
12	OS/2 Anwendung.
13	DOS 4.0 Anwendung.
14	Unbekannter .EXE-Typ.
15	Versuch, im Protected Mode (Standard- oder erweiterter 386-Modus) eine .EXE Datei zu laden, die für eine frühere Windows-Version erstellt wurde.
16	Versuch, eine zweite Instanz einer .EXE-Datei zu laden, die mehrere beschreibbare Datensegmente enthält.
17	Versuch, im Large-Frame-EMS-Modus eine zweite Instanz einer Anwendung zu laden, die bestimmte DLLs einbindet, die schon im Gebrauch sind, aber nicht gemeinsam genutzt werden können.
18	Versuch, im Real Mode eine Anwendung zu laden, die nur für den Protected Mode eingerichtet ist.

LoadMenu

Syntax

```
HMENU LoadMenu(HANDLE Instance, LPSTR MenuName);
```

Beschreibung Diese Funktion lädt die durch den Parameter *MenuName* bezeichnete Menüressource aus der ausführbaren Datei, die

dem durch den Parameter *Instance* bezeichneten Modul zugeordnet ist.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei das Menü enthält.

MenuName: Zeigt auf einen nullterminierten String, der den Namen der Menüressource enthält.

Rückgabewert Der Rückgabewert bezeichnet eine Menüressource, wenn die Funktion erfolgreich war, andernfalls ist er 0.

Hinweis Der Parameter *MenuName* kann auch einen Wert vom Typ *MakeIntResource* beinhalten. Ist dies der Fall, muß die ID im niederwertigen WORD von *MenuName* stehen und das höherwertige WORD muß auf 0 gesetzt werden.

LoadMenuIndirect

Syntax
```
HMENU LoadMenuIndirect(LPSTR MenuTemplate);
```

Beschreibung Diese Funktion lädt das Menü in den durch den Parameter *MenuTemplate* bezeichneten Speicher.

Die Vorlage, die von *MenuTemplate* angegeben wird, ist eine Kopfzeile, auf die eine Sammlung einer oder mehrerer *MENUITEMTEMPLATE*-Strukturen folgt, von denen jede einen oder mehrere Menüpunkte oder Pop-up-Menüs beinhalten kann.

MenuTemplate: Zeigt auf eine Menüvorlage, die eine Sammlung einer oder mehrerer *MENUITEMTEMPLATE*-Strukturen ist.

Rückgabewert Der Rückgabewert bezeichnet das Menü, wenn die Funktion erfolgreich war, andernfalls ist er 0.

LoadModule

Syntax
```
HANDLE LoadModule(LPSTR ModuleName,
LPVOID ParameterBlock);
```

Beschreibung Diese Funktion lädt ein Windows-Programm und führt es aus oder erzeugt eine neue Instanz eines existierenden Windows-Programms.

ModuleName: Zeigt auf einen nullterminierten String, der den Dateinamen der Anwendung enthält, die gestartet werden soll. Enthält *ModuleName* keinen Verzeichnispfad, sucht Windows in folgender Reihenfolge nach der ausführbaren Datei:

1. Im aktuellen Verzeichnis.

2. Im Windows-Verzeichnis (das Verzeichnis, in dem sich die Datei WIN.COM befindet); die Funktion *GetWindowsDirectory* liefert den Pfadnamen dieses Verzeichnisses.

3. Im Windows-Systemverzeichnis (das Verzeichnis, das Systemdateien wie KERNEL.EXE enthält); die Funktion *GetSystemDirectory* liefert den Pfadnamen dieses Verzeichnisses.

4. In den Verzeichnissen, die in der Environment-Variablen PATH aufgelistet sind.

5. In den Verzeichnissen, die in einem Netz im Befehl MAP angegeben wurden.

Wenn der Dateiname der Anwendung keine Namenserweiterung enthält, wird von .EXE ausgegangen.

ParameterBlock: Zeigt auf eine Datenstruktur, die aus vier Feldern besteht, die einen Parameterblock definieren. Die Datenstruktur besteht aus den folgenden Feldern:

Feld	Beschreibung
EnvSeg (WORD)	Bestimmt die Segmentadresse der Umgebung, unter der das Modul laufen soll; 0 zeigt an, daß die Windows-Umgebung kopiert werden soll.
CmdLine (DWORD)	Zeigt auf einen nullterminierten String, der eine korrekt gebildete Kommandozeile enthält. Dieser String darf eine Länge von 120 Bytes nicht überschreiten.
CmdShow (DWORD)	Zeigt auf eine Datenstruktur, die zwei Werte in Word-Länge enthält. Der erste Wert muß immer auf zwei gesetzt werden. Der zweite Wert gibt an, wie das Anwendungsfenster dargestellt werden soll. In Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »SW_XXXX« findet sich eine Beschreibung der möglichen Werte.
Reserved (DWORD)	Ist reserviert und muß 0 sein. Alle ungenutzten Felder sollten auf 0 gesetzt werden, mit Ausnahme von *CmdLine*, das auf einen Nullstring zeigen muß, wenn es nicht benötigt wird.

Rückgabewert

Der Rückgabewert bezeichnet bei erfolgreicher Ausführung die Instanz des geladenen Moduls. Andernfalls hat er einen Wert kleiner als 32, der den Fehler bezeichnet. Die folgende Liste beschreibt die Fehlerwerte, die zurückgeliefert werden:

Wert	Bedeutung
0	Zu wenig Speicher.
2	Datei nicht gefunden.
3	Pfad nicht gefunden.
5	Versuch, eine Task dynamisch einzubinden.
6	Bibliothek benötigt getrennte Datensegmente für jede Task.

Wert	Bedeutung
10	Falsche Windows-Version.
11	Ungültige .EXE Datei (Keine Windows-.EXE oder Fehler in der .EXE).
12	OS/2 Anwendung.
13	DOS 4.0 Anwendung.
14	Unbekannter .EXE-Typ.
15	Versuch, im Protected Mode (Standard- oder erweiterter 386-Modus) eine .EXE Datei zu laden, die für eine frühere Windows-Version erstellt wurde.
16	Versuch, eine zweite Instanz einer .EXE-Datei zu laden, die mehrere beschreibbare Datensegmente enthält.
17	Versuch, im Large-Frame-EMS-Modus eine zweite Instanz einer Anwendung zu laden, die bestimmte DLLs (dynamische Linkbibliotheken) einbindet, die schon im Gebrauch sind, aber nicht gemeinsam genutzt werden können.
18	Versuch, im Real Mode eine Anwendung zu laden, die nur im Protected Mode ablauffähig ist.

Hinweis Die Funktion *WinExec* stellt eine alternative Methode dar, ein Programm auszuführen.

Querverweis *WinExec*

LoadResource

Syntax
```
HANDLE LoadResource(HANDLE Instance, HANDLE ResInfo);
```

Beschreibung Diese Funktion lädt eine durch den Parameter *ResInfo* bezeichnete Ressource aus der ausführbaren Datei, die dem durch den Parameter *Instance* bezeichneten Modul zugeordnet ist.

LoadResource lädt die Ressource nur in den Speicher, wenn sie nicht schon vorher geladen wurde. Andernfalls liefert sie ein Handle zur bestehenden Ressource.

Instance: Bezeichnet eine Instanz des Moduls, dessen ausführbare Datei die Ressource enthält.

ResInfo: Bezeichnet die gewünschte Ressource. Voraussetzung ist, daß dieses Handle durch die Funktion *FindResource* erzeugt wurde.

Rückgabewert Der Rückgabewert bezeichnet den globalen Speicherblock, der die Daten empfängt, die der Ressource zugeordnet sind. Er ist 0, wenn die Ressource nicht existiert.

Hinweis Die Ressource wird erst geladen, wenn die Funktion *LockResource* aufgerufen wird, um das Handle, das von *LoadResource* zurückgeliefert wurde, in einen **far**-Zeiger auf die Ressourcedaten zu übersetzen.

LoadString

Syntax
```
int LoadString(HANDLE Instance, WORD ID, LPSTR Buffer,
int BufferMax);
```

Beschreibung Diese Funktion lädt eine String-Ressource, die durch den Parameter *ID* bezeichnet wird, aus der ausführbaren Datei, die dem Modul zugeordnet ist, das durch den Parameter *Instance* vorgegeben wird. Die Funktion kopiert den String in den Puffer, auf den der Parameter *Buffer* zeigt, und fügt ein abschließendes Nullzeichen an.

Instance: Bezeichnet eine Instanz des Moduls, dessen ausführbare Datei die String-Ressource enthält.

ID: Bestimmt den Integerbezeichner des Strings, der geladen werden soll.

Buffer: Zeigt auf den Puffer, der den String aufnehmen soll.

BufferMax: Bestimmt die maximale Anzahl der Zeichen, die in den Puffer kopiert werden sollen (mit Nullzeichen). Der String wird abgeschnitten, wenn er länger als die angegebene Zeichenanzahl ist.

Rückgabewert Der Rückgabewert gibt die tatsächlich in den Puffer kopierte Anzahl von Zeichen an (ohne Nullzeichen). Er ist 0, wenn die String-Ressource nicht existiert oder wenn der String leer ist.

LOBYTE

Syntax
```
BYTE LOBYTE(int Integer);
```

Beschreibung Dieses Makro liefert das niederwertige Byte des Integerwertes, der durch den Parameter *Integer* bestimmt wird.

Integer: Bestimmt den Wert, der umgewandelt werden soll.

Rückgabewert Der Rückgabewert bestimmt das niederwertige Byte des Wertes.

LocalAlloc

Syntax
```
HANDLE LocalAlloc(WORD Flags, WORD Bytes);
```

Beschreibung Diese Funktion reserviert die Anzahl von Bytes im lokalen Heap, die im Parameter *Bytes* angegeben wird. Der Speicherblock kann entweder fest oder verschiebbar sein. Dies wird im Parameter *Flags* angegeben.

Flags: Bestimmt, wie der Speicher reserviert werden soll. Er kann einen oder mehrere der folgenden Werte annehmen:

Wert	Bedeutung
LMEM_DISCARDABLE	Reserviert Speicher und ordnet ihm das Attribut »discardable« zu. Kann nur mit *LMEM_MOVEABLE* benutzt werden.
LMEM_FIXED	Reserviert festen Speicher.
LMEM_MODIFY	Modifiziert das Flag *LMEM_DISCARDABLE*. Kann nur mit *LMEM_DISCARDABLE* benutzt werden.
LMEM_MOVEABLE	Reserviert verschiebbaren Speicher. Kann nicht mit *LMEM_FIXED* benutzt werden.
LMEM_NOCOMPACT	Verdichtet oder verwirft keinen Speicher, um die gewünschte Reservierung durchzuführen.
LMEM_NODISCARD	Verwirft keinen Speicher, um die gewünschte Reservierung durchzuführen.
LMEM_ZEROINIT	Initialisiert den Speicherinhalt mit 0.

Nach Auswahl von *LMEM_FIXED* oder *LMEM_MOVEABLE* werden die benötigten anderen Flags unter Verwendung des bitweisen ODER-Operators (&) kombiniert.

Bytes: Bestimmt die Anzahl von Bytes, die reserviert werden sollen.

Rückgabewert Der Rückgabewert bezeichnet den neu reservierten lokalen Speicherblock, wenn die Funktion erfolgreich war, andernfalls ist er 0.

Hinweis Ist das Datensegment, das den Heap enthält, verschiebbar, so veranlaßt der Aufruf dieser Funktion eine Verschiebung des Segments, wenn Windows den Heap vergrößern muß und die Größe des Heaps in der gegenwärtigen Position nicht vergrößert werden kann.

Eine Anwendung kann Windows daran hindern, das Datensegment zu verlagern, indem sie die Funktion *LockData* aufruft, um das Datensegment zu sperren.

Wurde die Funktion erfolgreich ausgeführt, reserviert sie mindestens die gewünschte Byteanzahl. Der tatsächlich reservierte Speicherplatz kann größer sein. Um den wirklich reservierten Speicher zu ermitteln, kann die Funktion *LocalSize* aufgerufen werden.

Querverweis *LockData*

LocalCompact

Syntax `WORD LocalCompact(WORD MinFree);`

Beschreibung Diese Funktion stellt die die vom Parameter *MinFree* angegebe-

ne Anzahl an freien Bytes des Speichers zur Verfügung, indem sie den lokalen Heap des Moduls – falls nötig – verdichtet.

Die Funktion überprüft den lokalen Heap auf die angegebene Anzahl an zusammenhängenden freien Bytes. Wird kein Bereich dieser Größe gefunden, verdichtet die Funktion *LocalCompact* den lokalen Speicher, indem sie zuerst alle ungesperrten Blöcke in den oberen Speicher verschiebt.

Falls dadurch der angeforderte Speicher immer noch nicht zur Verfügung gestellt werden kann, verwirft die Funktion nach Möglichkeit solange nicht gesperrte und verschiebbare Speicherblöcke mit dem Attribut »discardable«, bis der angeforderte Speicher zur Verfügung steht.

MinFree: Bestimmt die Anzahl gewünschter freier Bytes. Wenn *MinFree* 0 ist, liefert die Funktion den Wert des größten zusammenhängenden Blocks freien Speichers zurück, verdichtet aber den Speicher nicht.

Rückgabewert Der Rückgabewert gibt die Anzahl der Bytes im größten freien Block des lokalen Speichers an.

LocalDiscard

Syntax
```
HANDLE LocalDiscard(HANDLE Mem);
```

Beschreibung Diese Funktion verwirft den lokalen Speicherblock, der durch den Parameter *Mem* bestimmt wird.

Der Sperrenzähler des Speicherblocks muß 0 sein. Der lokale Speicherblock wird zwar aus dem Speicher entfernt, aber sein Handle bleibt gültig. Eine Anwendung kann anschließend das Handle an die Funktion *LocalReAlloc* übergeben, um einen weiteren lokalen Speicherblock zu reservieren, der durch dieses Handle bezeichnet wird.

Mem: Bezeichnet den lokalen Speicherblock, der verworfen werden soll.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war, andernfalls ist er identisch mit *Mem*.

LocalFlags

Syntax
```
WORD LocalFlags(HANDLE Mem);
```

Beschreibung Diese Funktion liefert Informationen über den bezeichneten lokalen Speicherblock.

Mem: Bezeichnet den lokalen Speicherblock.

Rückgabewert Der Rückgabewert enthält eines der folgenden Speicherreservierungs-Flags im oberen Byte.

Wert	Bedeutung
LMEM_DISCARDABLE	Block hat das Attribut »discardable«.
LMEM_DISCARDED	Block wurde verworfen.

Das untere Byte des Rückgabewertes enthält den Referenzzähler des Blocks. Mit Hilfe der Maske *LMEM_LOCKCOUNT* kann der Wert des Sperrenzählers aus dem Rückgabewert ermittelt werden.

LocalFree

Syntax
```
HANDLE LocalFree(HANDLE Mem);
```

Beschreibung Diese Funktion gibt den lokalen Speicherblock frei, der durch den Parameter *Mem* bezeichnet wird, und macht das Handle des Speicherblocks ungültig.

Mem: Bezeichnet den lokalen Speicherblock, der freigegeben werden soll.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war, andernfalls ist er identisch mit *Mem*.

LocalHandle

Syntax
```
HANDLE LocalHandle(WORD Mem);
```

Beschreibung Diese Funktion liefert das Handle des lokalen Speicherobjektes, dessen Adresse durch *Mem* bestimmt wird.

Mem: Bestimmt die Adresse eines lokalen Speicherobjektes.

Rückgabewert Der Rückgabewert bezeichnet das lokale Speicherobjekt.

LocalInit

Syntax
```
BOOL LocalInit(WORD Segment, WORD Start, WORD End);
```

Beschreibung Diese Funktion initialisiert einen lokalen Heap in dem Segment, das durch den Parameter *Segment* angegeben wird.

Segment: Bestimmt die Segmentadresse des Segments, das den lokalen Heap enthalten soll.

Start: Bestimmt die Anfangsadresse des lokalen Heap innerhalb des Segments.

End: Bestimmt die Endadresse des lokalen Heap innerhalb des Segments.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der Heap initialisiert wurde, andernfalls ist er 0.

Hinweis

Ist der Parameter *Start* 0, so gibt der Parameter *End* den Abstand des letzten Bytes des lokalen Heap zu diesem Ausgangspunkt zurück: um zum Beispiel einen Heap von 4096 Byte mit dem ersten Byte bei Byte 0 zu initialisieren, müssen Sie *Start* auf 0 und *End* auf 4095 setzen.

LocalInit ruft die Funktion *GlobalLock* für das Datensegment auf, das den lokalen Heap enthält und stellt so sicher, daß das Datensegment nicht im Speicher verschoben wird.

Der Speicher wird verschoben, wenn folgenden Bedingungen erfüllt sind:

1. Das Datensegment ist verschiebbar.

2. Die Anwendung ruft eine der Funktionen *LocalAlloc* oder *LocalReAlloc* auf, und Windows muß infolgedessen den Heap vergrößern. Kann Windows das Datensegment, das den lokalen Heap enthält, nicht ohne Verschiebung vergrößern, so wird es verschoben.

Eine Anwendung kann Windows ausdrücklich daran hindern, das Datensegment zu verschieben, indem sie die Funktion *LockData* aufruft, um das Datensegment zu sperren. Sie kann diese Sperre entfernen, indem sie *UnlockData* aufruft.

LocalLock

Syntax

```
char NEAR * LocalLock(HANDLE Mem);
```

Beschreibung

Diese Funktion sperrt den lokalen Speicherblock, der durch den Parameter *Mem* bestimmt wird.

Der Block wird im Speicher an der gegebenen Adresse gesperrt und sein Referenzzähler um 1 erhöht. Gesperrter Speicher kann nicht verschoben oder verworfen werden. Der Block bleibt im Speicher gesperrt, bis sein Referenzzähler durch Verwendung der Funktion *LocalUnlock* auf 0 herabgesetzt wird.

Mem: Bezeichnet den lokalen Speicherblock, der gesperrt werden soll.

Rückgabewert

Wenn die Funktion erfolgreich ausgeführt wurde, zeigt der Rückgabewert auf das erste Byte im Speicher des lokalen Blocks, andernfalls ist er NULL.

LocalReAlloc

Syntax

```
HANLDE LocalReAlloc(HANLDE Mem, WORD Bytes, WORD Flags);
```

Beschreibung

Diese Funktion vergrößert oder verkleinert den lokalen Speicherblock, der durch den Parameter *Mem* bestimmt wird, auf die Anzahl von Bytes, die im Parameter *Bytes* vorgegeben ist oder sie ändert die Attribute des angegebenen Speicherblocks.

Mem: Bezeichnet den lokalen Speicherblock, der verändert werden soll.

Bytes: Bestimmt die neue Größe des Speicherblocks.

Flags: Bestimmt, wie der lokale Speicherblock verändert werden soll. Dieser Parameter kann einen oder mehrere der folgenden Werte annehmen:

Wert	Bedeutung
LMEM_DISCARDABLE	Speicher wird mit dem Attribut »discardable« versehen. Dieses Flag kann nur zusammen mit *LMEM_MODIFY* benutzt werden.
LMEM_MODIFY	Speicherflags werden modifiziert. Der Parameter *Bytes* wird ignoriert. Dieses Flag kann nur zusammen mit *LMEM_DISCARDABLE* benutzt werden.
LMEM_MOVEABLE	Speicher ist verschiebbar. Wenn *Bytes* 0 ist, verursacht dieses Flag die Freigabe eines vorher festen Blocks, wenn der Referenzzähler des Blocks 0 ist, das Verwerfen eines verschiebbaren Objektes. Wenn *Bytes* ungleich 0 und der Block, der von *Mem* bezeichnet wird, fest ist, erlaubt dieses Flag die Verschiebung des veränderten Blocks an eine neue feste Position. Beachten Sie, daß das Handle, das von der Funktion *LocalReAlloc* zurückgeliefert wird, in diesem Fall von dem Handle, das der Funktion übergeben wurde, verschieden sein kann. Dieses Flag kann nicht mit *LMEM_MODIFIY* benutzt werden.
LMEM_NOCOMPACT	Speicher wird nicht verdichtet oder verworfen, um die gewünschte Reservierung sicherzustellen. Dieses Flag kann nicht mit *LMEM_MODIFY* benutzt werden.
LMEM_NODISCARD	Objekte werden nicht verworfen, um die gewünschte Reservierung sicherzustellen. Dieses Flag kann nicht mit *LMEM_MODIFY* benutzt werden.
LMEM_ZEROINIT	Wenn der Block größer wird, werden die zusätzlichen Speicherinhalte mit 0 initialisiert. Dieses Flag kann nicht mit *LMEM_MODIFY* benutzt werden.

Rückgabewert Der Rückgabewert bezeichnet den geänderten lokalen Speicher, wenn die Funktion erfolgreich war. Er ist 0, wenn die

Größe des lokalen Speicherblocks nicht verändert werden konnte. Der Rückgabewert ist immer mit dem Parameter *Mem* identisch, wenn nicht das Flag *LMEM_MOVEABLE* benutzt wird, um die Verschiebung eines festen Speicherblocks an eine neue Speicherstelle zu erlauben.

Hinweis Ist das Datensegment, das den Heap enthält, verschiebbar, veranlaßt der Aufruf dieser Funktion eine Verschiebung des Segments, wenn Windows den Heap vergrößern muß und die Größe des Heap in der gegenwärtigen Position nicht vergrössern kann.

Eine Anwendung kann Windows daran hindern, das Datensegment zu verschieben, indem sie das Makro *LockData* aufruft, um das Datensegment zu sperren.

Querverweis *LockData*

LocalShrink

Syntax
```
WORD LocalShrink(HANDLE Seg, WORD Size);
```

Beschreibung Diese Funktion verkleinert den angegebenen Heap auf die Größe, die vom Parameter *Size* angegeben wird. Die kleinste mögliche Größe des automatischen lokalen Heap ist in der Moduldefinitionsdatei der Anwendung festgelegt.

Seg: Bezeichnet das Segment, welches den lokalen Heap enthält.

Size: Bestimmt die Größe des lokalen Heap in Bytes nach der erwünschten Verkleinerung.

Rückgabewert Der Rückgabewert gibt die Größe des lokalen Heap nach der Verkleinerung an.

Hinweis Wenn *Seg* 0 ist, verkleinert die Funktion *LocalShrink* den lokalen Heap im gegenwärtigen Datensegment. Windows wird den Teil des Datensegments nicht verkleinern, der den Stack und die statischen Variablen enthält.

Verwenden Sie die Funktion *GlobalSize*, um die neue Größe des Datensegments zu bestimmen.

Querverweis *GlobalSize*

LocalSize

Syntax
```
WORD LocalSize(HANDLE Mem);
```

Beschreibung Diese Funktion liefert die gegenwärtige Größe des lokalen Speicherblocks, der vom Parameter *Mem* bezeichnet wird.

Mem: Bezeichnet den lokalen Speicherblock.

Rückgabewert Der Rückgabewert gibt die Größe des angegebenen Speicherblocks in Bytes an. Er ist 0, wenn das vorgegebene Handle ungültig ist.

Hinweis Die tatsächliche Größe eines Speicherblocks ist manchmal größer, als bei der Reservierung angefordert wurde.

LocalUnlock

Syntax
```
BOOL LocalUnlock(HANDLE Mem);
```

Beschreibung Diese Funktion macht die Sperrung des durch den Parameter *Mem* bezeichneten lokalen Speicherblocks rückgängig und verringert den Referenzzähler des Blocks um 1.

Der Block ist ungesperrt und kann verschoben und verworfen werden, wenn der Referenzzähler 0 ist.

Mem: Bezeichnet den lokalen Speicherblock, dessen Sperrung aufgehoben werden soll.

Rückgabewert Der Rückgabewert ist Null, wenn der Referenzzähler des Blocks auf 0 verringert wurde, andernfalls ist er ungleich Null.

LockData

Syntax
```
HANDLE LockData(int Dummy);
```

Beschreibung Dieses Makro sperrt das aktuelle Datensegment im Speicher. Es ist für Module bestimmt, die verschiebbare Datensegmente haben.

Dummy: Wird nicht verwendet. Er sollte auf 0 gesetzt werden.

Rückgabewert Der Rückgabewert bezeichnet das gesperrte Datensegment, wenn die Funktion erfolgreich war, andernfalls ist er 0.

Querverweis *UnlockData*

LockResource

Syntax
```
LPSTR LockResource(HANDLE ResData);
```

Beschreibung Diese Funktion ermittelt die absolute Speicheradresse der durch den Parameter *ResData* bezeichneten geladenen Ressource. Die Ressource wird im Speicher gesperrt, und ihr Referenzzähler wird um 1 erhöht. Die gesperrte Ressource kann nicht verschoben oder verworfen werden. Die Ressource bleibt solange im Speicher gesperrt, bis ihr Referenzzähler durch Aufrufe der Funktion *FreeResource* auf 0 verringert wurde.

Wenn die durch den Parameter *ResData* bezeichnete Ressource verworfen wurde, wird die mit ihr verbundene Ressour-

ce-Handle-Funktion (falls vorhanden), aufgerufen, bevor die Funktion *LockResource* zurückkehrt. Die Ressource-Handle-Funktion kann – falls gewünscht – die Ressource erneut berechnen und wieder laden. Nachdem die Ressource-Handle-Funktion zurückgekehrt ist, macht *LockResource* einen weiteren Versuch, die Ressource zu sperren und liefert das Resultat als Rückgabewert.

ResData: Bezeichnet die gewünschte Ressource. Es wird vorausgesetzt, daß dieses Handle durch die Funktion *LoadResource* erzeugt wurde.

Rückgabewert Der Rückgabewert zeigt auf das erste Byte der geladenen Ressource, falls sie gesperrt wurde, andernfalls ist er NULL.

Hinweis Die Verwendung des durch die Funktion *FindResource* zurückgelieferten Handle als Parameter *ResData* verursacht einen Fehler. Verwenden Sie die Funktion *UnlockResource*, um die Sperrung einer Ressource rückgängig zu machen, die durch *LockResource* gesperrt wurde.

LockSegment

Syntax
```
HANDLE LockSegment(WORD Segment);
```

Beschreibung Diese Funktion sperrt das Segment, dessen Segmentadresse durch den Parameter *Segment* bestimmt wird. Wenn *Segment* -1 ist, sperrt die Funktion *LockSegment* das aktuelle Datensegment. Mit Ausnahme von Segmenten im Protected Mode (Standard- oder erweiterter 386-Modus), die nicht das Attribut »discardable« haben, wird das Segment an der gegebenen Adresse im Speicher gesperrt und sein Sperrenzähler um 1 erhöht. Gesperrter Speicher wird nicht verschoben oder verworfen, außer wenn ein Teil des Segments durch die Funktion *GlobalReAlloc* neu belegt wurde. Das Segment bleibt im Speicher gesperrt, bis sein Sperrenzähler auf 0 verringert wird. Im Protected Mode erhöht *LockSegment* den Sperrenzähler ausschließlich für automatische Datensegmente mit dem Attribut »discardable«.

Jedesmal, wenn eine Anwendung *LockSegment* für ein Segment aufruft, muß sie entsprechend *UnlockSegment* für das Segment aufrufen. Die Funktion *UnlockSegment* verringert den Sperrenzähler des Segments.

Andere Funktionen können ebenfalls den Sperrenzähler eines Speicherobjekts beeinflussen. In der Beschreibung der Funktion *GlobalFlags* findet sich eine Liste dieser Funktionen.

Segment: Bestimmt die Segmentadresse des Segments, das gesperrt werden soll. Wenn *Segment* -1 ist, sperrt die Funktion *LockSegment* das aktuelle Datensegment.

Rückgabewert Der Rückgabewert bezeichnet das Datensegment, wenn die Funktion erfolgreich war. Wenn das Objekt verworfen wurde oder ein Fehler aufgetreten ist, ist der Rückgabewert NULL.

Querverweis *UnlockSegment*

_lopen

Syntax
```
int _lopen(LPSTR PathName, int ReadWrite);
```

Beschreibung Diese Funktion öffnet die Datei mit dem durch den Parameter *PathName* festgelegten Namen. Der Parameter *ReadWrite* legt den Zugriffsmodus auf die Datei fest, wenn die Funktion sie öffnet. Wenn die Datei existiert und nur für Schreibaktionen geöffnet wird, verkürzt die Funktion die Dateilänge auf 0.

Der Dateizeiger wird beim Öffnen auf den Anfang der Datei gesetzt.

PathName: Zeigt auf einen nullterminierten String, der den Namen der zu öffnenden Datei enthält. Der String muß aus Zeichen des ANSI-Zeichensatzes bestehen.

ReadWrite: Bestimmt, ob die Funktion die Datei zum Lesezugriff, zum Schreibzugriff oder zu beidem öffnen soll. Der Parameter muß einen der folgenden Werte annehmen:

Wert	Bedeutung
OF_READ	Öffnet die Datei nur zum Lesen.
OF_READWRITE	Öffnet die Datei zum Lesen und zum Schreiben.
OF_SHARE_COMPAT	Öffnet die Datei im Kompatibilitätsmodus und erlaubt das Öffnen der Datei, sooft sie benötigt wird. *OpenFile* schlägt fehl, wenn die Datei mit einem anderen Modus zur Mehrfachbenutzung geöffnet wurde.
OF_SHARE_DENY_NONE	Öffnet die Datei, ohne anderen Prozessen Lese- oder Schreibzugriffe auf die Datei zu verweigern. *OpenFile* mißlingt, wenn die Datei durch einen anderen Prozeß im Kompatibilitätsmodus geöffnet wurde.
OF_SHARE_DENY_READ	Öffnet die Datei und verweigert anderen Prozessen den Lesezugriff auf die Datei. *OpenFile* mißlingt, wenn die Datei im Kompatibilitätsmodus oder zum Lesezugriff durch einen anderen Prozeß geöffnet wurde.

Wert	Bedeutung
OF_SHARE_DENY_WRITE	Öffnet die Datei und verweigert anderen Prozessen den Schreibzugriff auf die Datei. *OpenFile* mißlingt, wenn die Datei im Kompatibilitätsmodus oder zum Schreibzugriff durch irgendeinen anderen Prozeß geöffnet wurde.
OF_SHARE_EXCLUSIVE	Öffnet die Datei im Exklusivmodus und verweigert anderen Prozessen sowohl Lese- als auch Schreibzugriffe auf die Datei. *OpenFile* hat keinen Erfolg, wenn die Datei in einem anderen Modus für Lese- oder Schreibzugriffe geöffnet wurde; dies gilt auch für den aktuellen Prozeß.
OF_WRITE	Öffnet die Datei nur zum Schreiben.

Rückgabewert Der Rückgabewert liefert das DOS-Datei-Handle, wenn die Funktion die Datei geöffnet hat. Andernfalls ist der Rückgabewert -1.

LOWORD

Syntax
```
WORD LOWORD(DWORD AnInteger);
```

Beschreibung Dieses Makro liefert das niederwertige WORD aus dem 32-Bit-Integer, der durch den Parameter *AnInteger* bestimmt wird.

AnInteger: Bestimmt den Wert, der bearbeitet werden soll.

Rückgabewert Der Rückgabewert bestimmt das niederwertige WORD des 32-Bit-Integer.

LPtoDP

Syntax
```
BOOL LPtoDP(HDC DC, LPPOINT Points, int Count);
```

Beschreibung Die Funktion *LPtoDP* transformiert die Koordinaten jedes durch den Parameter *Points* bezeichneten Punktes vom virtuellen Koordinatensystem der GDI in ein Gerätekoordinatensystem. Die Transformation hängt vom aktuellen Abbildungsmodus ab.

DC: Bezeichnet den Gerätekontext.

Points: Zeigt auf ein Array von Punkten. Jeder Punkt im Array ist eine *POINT*-Datenstruktur.

Count: Bestimmt die Anzahl von Punkten im Array.

Rückgabewert Der Rückgabewert gibt an, ob alle Punkte umgewandelt sind. Er ist ungleich Null, wenn alle Punkte umgewandelt sind, andernfalls ist er Null.

_lread

Syntax
```
WORD _lread(int FileHandle, LPSTR Buffer, int Bytes);
```

Beschreibung Diese Funktion liest Daten aus der durch den Parameter *FileHandle* bestimmten Datei. Der Parameter *Bytes* bezeichnet die Anzahl der Bytes, die gelesen werden sollen. Die Funktion liefert die tatsächlich gelesene Anzahl von Bytes zurück. Wenn die Funktion versucht hat, die Datei am Dateiende zu lesen, ist der Rückgabewert 0.

FileHandle: Bestimmt das DOS-Datei-Handle der Datei, die gelesen werden soll.

Buffer: Zeigt auf einen Puffer, der die Daten aufnehmen soll, welche aus der Datei gelesen werden.

Bytes: Bestimmt die Anzahl der Bytes, die aus der Datei gelesen werden sollen.

Rückgabewert Der Rückgabewert liefert die Anzahl der Bytes, die die Funktion tatsächlich aus der Datei gelesen hat, oder -1, wenn die Funktion fehlschlägt.

Der Rückgabewert ist kleiner als *Bytes*, wenn die Funktion das Dateiende erreicht hat, bevor sie die angegebene Anzahl von Bytes lesen konnte.

lstrcat

Syntax
```
LPSTR lstrcat(LPSTR Str1, LPSTR Str2);
```

Beschreibung Diese Funktion hängt *Str2* an den String an, der durch *Str1* bezeichnet wird, schließt den entstehenden String mit einem Nullzeichen ab und liefert einen Zeiger auf den verbundenen String *Str1*.

Alle Strings müssen kleiner als 64 KByte sein.

Str1: Zeigt auf ein Zeichen-Array, das einen nullterminierten String enthält, an den die Funktion *Str2* hängen soll. Das Zeichen-Array, das den String enthält, muß groß genug sein, um beide Strings aufzunehmen.

Str2: Zeigt auf einen nullterminierten String, den die Funktion an *Str1* anhängen soll.

Rückgabewert Der Rückgabewert ist bei erfolgreicher Ausführung ein Zeiger auf *Str1*, andernfalls ist er 0.

lstrcmp

Syntax
```
int lstrcmp(LPSTR Str1, LPSTR Str2);
```

Beschreibung Diese Funktion vergleicht die beiden durch *Str1* und *Str2* bezeichneten Strings lexikographisch und liefert einen Wert zurück, der das Ergebnis des Vergleichs angibt.

Der Vergleich basiert auf der aktuellen Sprache, die vom Anwender im Setup oder in der Systemsteuerung ausgewählt wurde. Der Vergleich unterscheidet zwischen Groß- und Kleinschreibung.

Beide Strings müssen kleiner als 64 KByte sein.

Str1: Zeigt auf den ersten nullterminierten String, der verglichen werden soll.

Str2: Zeigt auf den zweiten nullterminierten String, der verglichen werden soll.

Rückgabewert Der Rückgabewert zeigt an, ob *Str1* kleiner, gleich oder größer als *Str2* ist. Dabei gilt:

<0	*Str1* ist kleiner als *Str2*.
$=0$	*Str1* ist mit *Str2* identisch.
>0	*Str1* ist größer als *Str2*.

lstrcmpi

Syntax
```
int lstrcmpi(LPSTR Str1, LPSTR Str2);
```

Beschreibung Diese Funktion vergleicht die beiden durch *Str1* und *Str2* bezeichneten Strings lexikographisch und liefert einen Wert zurück, der das Ergebnis des Vergleichs angibt.

Der Vergleich basiert auf der aktuellen Sprache, die vom Benutzer im Setup oder in der Systemsteuerung ausgewählt wurde. Zwischen Groß- und Kleinschreibung wird nicht unterschieden.

Alle Strings müssen kleiner als 64 KByte sein.

Str1: Zeigt auf den ersten nullterminierten String, der verglichen werden soll.

Str2: Zeigt auf den zweiten nullterminierten String, der verglichen werden soll.

Rückgabewert Der Rückgabewert zeigt an, ob *Str1* kleiner, gleich oder größer als *Str2* ist. Dabei gilt:

<0	*Str1* ist kleiner als *Str2*.
$=0$	*Str1* ist mit *Str2* identisch.
>0	*Str1* ist größer als *Str2*.

lstrcpy

Syntax `LPSTR lstrcpy(LPSTR Str1, LPSTR Str2);`

Beschreibung Diese Funktion kopiert *Str2* einschließlich des abschließenden Nullzeichens in den Bereich, der von *Str1* bestimmt wird und gibt *Str1* zurück. Alle Strings müssen kleiner als 64 KByte sein.

Str1: Zeigt auf einen Puffer, der den Inhalt von *Str2* aufnehmen soll. Der Puffer muß groß genug für *Str2* sein.

Str2: Zeigt auf den nullterminierten String, der kopiert werden soll.

Rückgabewert Der Rückgabewert ist ein Zeiger auf *Str1*. Er ist 0, wenn die Funktion fehlschlägt.

lstrlen

Syntax `int lstrlen(LPSTR Str);`

Beschreibung Diese Funktion gibt die Länge von *Str* in Bytes ohne das abschließende Nullzeichen zurück. Der String muß kleiner als 64 KByte sein.

Str: Zeigt auf einen nullterminierten String.

Rückgabewert Es wird die Länge von *Str* zurückgegeben. Im Fall eines Fehlers wird kein Wert zurückgeliefert.

_lwrite

Syntax `WORD _lwrite(int FileHandle, LPSTR Buffer, int Bytes);`

Beschreibung Diese Funktion schreibt Daten in die Datei, die durch den Parameter *FileHandle* bestimmt wird. Der Parameter *Bytes* legt die Anzahl von Bytes fest, die aus dem durch *Buffer* bezeichneten Puffer geschrieben werden sollen.

Der Rückgabewert der Funktion liefert die tatsächlich in die Datei geschriebene Anzahl von Bytes.

FileHandle: Bestimmt das DOS-Datei-Handle der Datei, in die geschrieben werden soll.

Buffer: Zeigt auf einen Puffer, der die Daten enthält, die in die Datei geschrieben werden sollen.

Bytes: Bestimmt die Anzahl von Bytes, die in die Datei geschrieben werden sollen.

Rückgabewert Es wird die Anzahl von Bytes zurückgeliefert, die die Funktion tatsächlich in die Datei geschrieben hat. Im Fall eines Fehlers ist der Rückgabewert -1.

Hinweis Der Puffer, der von *Buffer* bestimmt wird, darf nicht über das Ende eines Segments hinausreichen.

MAKELONG

Syntax
```
DWORD MAKELONG(WORD Low, WORD High);
```

Beschreibung Dieses Makro erzeugt einen vorzeichenlosen 32-Bit-Integer durch die Verkettung von zwei durch die Parameter *Low* und *High* bezeichneten Worten.

Low: Bestimmt das niederwertige WORD des neuen DWORD.

High: Bestimmt das höherwertige WORD des neuen DWORD.

Rückgabewert Der Rückgabewert ist der neue vorzeichenlose DWORD.

MakeProcInstance

Syntax
```
FARPROC MakeProcInstance(FARPROC Proc, HANDLE Instance);
```

Beschreibung Diese Funktion erzeugt die Adresse einer Prozedurinstanz.

Eine Prozedurinstanzadresse zeigt auf den Prologcode, der vor der Ausführung der Funktion abläuft. Der Prolog verknüpft das Datensegment der durch den Parameter *Instance* bezeichneten Instanz mit der Funktion, auf die der Parameter *Proc* zeigt. Wenn die Funktion ausgeführt wird, hat sie Zugriff auf Variablen und Daten im Datensegment dieser Instanz.

Proc: Ist die Adresse einer Prozedurinstanz.

Instance: Bezeichnet die Instanz, die dem gewünschten Datensegment zugeordnet ist.

Rückgabewert Der Rückgabewert zeigt bei erfolgreicher Ausführung auf die Funktion, andernfalls ist er 0.

Hinweis Die Funktion *MakeProcInstance* darf nur verwendet werden, um auf Funktionen von Instanzen des aktuellen Moduls zuzugreifen. Diese Funktion ist für Bibliotheksmodule nicht erforderlich.

Nachdem *MakeProcInstance* für eine bestimmte Funktion aufgerufen worden ist, sollten alle Aufrufe dieser Funktion über die gelieferte Adresse erfolgen.

MakeProcInstance erzeugt mehr als eine Prozedurinstanz. Eine Anwendung sollte *MakeProcInstance* nicht öfter als einmal mit derselben Funktion und demselben Instanz-Handle benutzen, damit nicht Speicherplatz vergeudet wird.

Um ein Datensegment mit einer Funktion zu verknüpfen, muß die Funktion durch eine EXPORTS-Anweisung in der Moduldefinitionsdatei exportiert werden.

MapDialogRect

Syntax `void MapDialogRect(HWND Dlg, LPRECT Rect);`

Beschreibung Diese Funktion konvertiert die in der *RECT*-Struktur angegebenen Dialogfenstereinheiten in Bildschirmeinheiten. Dialogfenstereinheiten sind in Ausdrücken der aktuellen Dialog-Basiseinheit festgelegt, die sich aus der durchschnittlichen Höhe und Breite der Zeichen in der Systemschrift ergibt. Eine horizontale Einheit ist ein Viertel der Dialog-Basisbreiteneinheit, eine vertikale Einheit ein Achtel der Dialog-Basishöheneinheit.

Die Funktion *GetDialogBaseUnits* gibt die Dialog-Basiseinheiten in Pixel zurück. *MapDialogRect* ersetzt die Dialogfenstereinheiten in *RECT* durch Bildschirmeinheiten (Pixel), so daß das Rechteck benutzt werden kann, um ein Dialogfenster zu erstellen oder ein Steuerelement innerhalb eines Fensters zu positionieren.

Dlg: Bezeichnet ein Dialogfenster.

Rect: Zeigt auf eine *RECT*-Datenstruktur, welche die zu konvertierenden Dialogfensterkoordinaten enthält.

Rückgabewert Keiner.

Hinweis Das Handle in *Dlg* muß mit einer der Funktionen *CreateDialog* oder *DialogBox* erzeugt werden.

Querverweis *GetDialogBaseUnits*

MapVirtualKey

Syntax `WORD MapVirtualKey(WORD Code, WORD MapType);`

Beschreibung Diese Funktion nimmt einen virtuellen Tastencode oder einen Scancode für eine Taste entgegen und gibt den zugehörigen Scancode, virtuellen Tastencode oder ASCII-Wert zurück.

Der Wert des Parameters *MapType* legt den Typ der Konvertierung fest, den diese Funktion ausführt.

Code: Bestimmt den virtuellen Tastencode oder Scancode für eine Taste. Die Interpretation des Parameters *Code* hängt vom Wert des Parameters *MapType* ab.

MapType: Bestimmt den Typ der Konvertierung, die ausgeführt werden soll. Der Parameter *MapType* kann einen der in der folgenden Werte beschriebenen annehmen:

Andere Werte sind reserviert und sollten nicht übergeben werden.

Wert	Bedeutung
0	Der Parameter *Code* bestimmt einen virtuellen Tastencode, die Funktion gibt den zugehörigen Scancode zurück
1	Der Parameter *Code* bestimmt einen Scancode, die Funktion gibt den zugehörigen virtuellen Tastencode zurück.
2	Der Parameter *Code* bestimmt einen virtuellen Tastencode, die Funktion gibt den zugehörigen ASCII-Wert zurück, der gilt, wenn die Shift-Taste nicht gedrückt ist.

Rückgabewert Der Rückgabewert hängt vom Wert der Parameter *Code* und *MapType* ab.

MessageBeep

Syntax
```
void MessageBeep(WORD BeepType);
```

Beschreibung Die Funktion erzeugt einen Warnton am Systemlautsprecher.

BeepType: Wird nicht verwendet und sollte auf 0 gesetzt werden.

Rückgabewert Keiner.

MessageBox

Syntax
```
int MessageBox(HWND Parent, LPSTR Txt, LPSTR Caption,
WORD TextType);
```

Beschreibung Diese Funktion erzeugt ein Fenster und stellt es mit einer von der Anwendung bereitgestellten Meldung und Kopfzeile dar. Hinzu kommt eine Kombination der vordefinierten Symbole und Schalter, die im Hinweis beschrieben sind.

Parent: Bezeichnet das Fenster, dem das Meldungsfenster gehört.

Txt: Zeigt auf einen nullterminierten String, der die zu zeigende Meldung enthält.

Caption: Zeigt auf einen nullterminierten String, der als Überschrift des Dialogfensters verwendet wird. Wenn der Parameter *Caption* 0 ist, wird die Standardüberschrift verwendet.

TextType: Bestimmt den Inhalt des Dialogfensters. Er kann jede Kombination der im Hinweis beschriebenen Werte sein, verknüpft durch den bitweisen ODER-Operator (&).

Rückgabewert Wenn zum Erzeugen des Dialogfensters der Speicherplatz nicht ausreicht, wird 0 zurückgegeben. Andernfalls ist der Rückgabewert einer der folgenden Menüoptionswerte des Dialogfensters:

Wert	Bedeutung
IDABORT	Aktionsschalter *Beenden* gedrückt.
IDCANCEL	Aktionsschalter *Abbrechen* gedrückt.
IDIGNORE	Aktionsschalter *Ignorieren* gedrückt.
IDNO	Aktionsschalter *Nein* gedrückt.
IDOK	Aktionsschalter *OK* gedrückt.
IDRETRY	Aktionsschalter *Wiederholen* gedrückt.
IDYES	Aktionsschalter *Ja* gedrückt.

Wenn das Meldungsfenster über einen Aktionsschalter *Abbrechen* verfügt, wird der Wert *IDCANCEL* zurückgegeben, wenn entweder die Taste Esc oder der Aktionsschalter *Abbrechen* gedrückt wird. Wenn das Meldungsfenster über keinen Aktionsschalter *Abbrechen* verfügt, hat das Drücken der Taste Esc keine Wirkung.

Hinweis

Wenn ein systemmodales Meldungsfenster erzeugt wird, um anzuzeigen, daß im System der Speicherplatz knapp wird, sollten die Strings, die als die Parameter *Txt* und *Caption* übergeben werden, nicht einer Ressourcendatei entnommen werden, da ein Versuch, die Ressource zu laden, fehlschlagen könnte.

Wenn eine Anwendung die Funktion *MessageBox* aufruft und die Flags *MB_ICONHAND* und *MB_SYSTEMMODAL* für den Parameter *TextType* vorgibt, stellt Windows das resultierende Meldungsfenster ungeachtet des verfügbaren Speicherplatzes dar. Wenn diese Flags gesetzt sind, wird die Länge des Meldungsfenstertextes auf eine Zeile begrenzt.

Wenn ein Meldungsfenster erzeugt wird, während ein Dialogfenster präsent ist, sollte das Handle des Dialogfensters als Parameter *Parent* verwendet werden. Dieser Parameter sollte kein untergeordnetes Fenster, wie etwa das Steuerelement eines Dialogfensters, bezeichnen.

Wert	Bedeutung
MB_ABORTRETRYIGNORE	Meldungsfenster enthält drei Schalter: *Beenden*, *Wiederholen* und *Ignorieren*.
MB_APPLMODAL	Der Anwender muß auf das Meldungsfenster reagieren, bevor er die Arbeit in dem durch den Parameter *Parent* bezeichneten Fenster fortsetzt. Der Anwender kann sich jedoch zu den Fenstern anderer Anwendungen bewegen und in diesen Fenstern arbeiten. *MB_APPLMODAL* ist der Standardwert, wenn weder *MB_SYSTEMMODAL* noch *MB_TASKMODAL* angegeben ist.

Wert	Bedeutung
MB_DEFBUTTON1	Erster Schalter ist der voreingestellte. Das heißt, Drücken von Return oder Leerzeichen hat die gleiche Wirkung wie das Betätigen des ersten Aktionsschalters. Der erste Schalter ist immer dann die Voreinstellung, wenn nicht *MB_DEFBUTTON2* bzw. *MB_DEFBUTTON3* angegeben ist.
MB_DEFBUTTON2	Zweiter Schalter ist der voreingestellte.
MB_DEFBUTTON3	Dritter Schalter ist der voreingestellte.
MB_ICONASTERISK	Gleichbedeutend mit *MB_ICONINFORMATION*.
MB_ICONEXCLAMATION	Ein Ausrufezeichensymbol erscheint im Meldungsfenster.
MB_ICONHAND	Gleichbedeutend mit *MB_ICONSTOP*.
MB_ICONINFORMATION	Ein Symbol, bestehend aus einem kleinen *i* in einem Kreis, erscheint im Meldungsfenster.
MB_ICONQUESTION	Ein Fragezeichensymbol erscheint im Meldungsfenster.
MB_ICONSTOP	Ein Stopzeichensymbol erscheint im Meldungsfenster.
MB_OK	Meldungsfenster enthält einen Schalter: *Ok*.
MB_OKCANCEL	Meldungsfenster enthält zwei Schalter: *Ok* und *Abbrechen*.
MB_RETRYCANCEL	Meldungsfenster enthält zwei Schalter: *Wiederholen* und *Abbrechen*.
MB_SYSTEMMODAL	Alle Anwendungen werden unterbrochen, bis der Anwender auf das Meldungsfenster reagiert. Wenn die Anwendung nicht *MB_ICONHAND* vorgibt, wird das Meldungsfenster erst modal, nachdem es erstellt worden ist; folglich empfangen das übergeordnete und andere Fenster weiterhin Botschaften, die sich aus seiner Aktivierung ergeben. Systemmodale Meldungsfenster dienen dazu, den Anwender von ernsten, möglicherweise schadensverursachenden Fehlern zu benachrichtigen, die sofortige Aufmerksamkeit erfordern (beispielsweise Speicherüberlauf).

Wert	Bedeutung
MB_TASKMODAL	Gleichbedeutend mit *MB_APPLMODAL* außer daß alle Hauptfenster, die zur aktuellen Task gehören, gesperrt werden, wenn der Parameter *Parent* 0 ist. Dieses Flag sollte verwendet werden, wenn die aufrufende Anwendung oder Bibliothek kein verfügbares Fenster-Handle hat, aber eines benötigt, um Eingaben an andere Fenster der aktuellen Anwendung zu vermeiden, ohne andere Anwendungen zu unterbrechen.
MB_YESNO	Meldungsfenster enthält zwei Schalter: *Ja* und *Nein*.
MB_YESNOCANCEL	Meldungsfenster enthält drei Schalter: *Ja*, *Nein* und *Abbrechen*.

ModifyMenu

Syntax

```
BOOL ModifyMenu(HMENU Menu, WORD Position, WORD Flags,
WORD IDNewItem, LPSTR NewItem);
```

Beschreibung Diese Funktion ändert eine vorhandene Menüoption an der Position, die durch den Parameter *Position* bestimmt wird. Die Anwendung gibt den neuen Status der Menüoption vor, indem sie die Werte im Parameter *Flags* einstellt.

Wenn diese Funktion ein der Menüoption zugeordnetes Pop-up-Menü ersetzt, entfernt sie das alte Pop-up-Menü und gibt den Speicher frei, der vom Pop-up-Menü benutzt wurde.

Menu: Bezeichnet das Menü, das verändert werden soll.

Position: Bestimmt die zu verändernde Menüoption. Die Interpretation des Parameters *Position* hängt von der Einstellung des Parameters *Flags* ab.

Flags	Position:
MF_BYPOSITION	Bestimmt die Position einer existierenden Menüoption. Die erste Option im Menü ist an Position 0.
MF_BYCOMMAND	Bestimmt die Befehls-ID der vorhandenen Menüoption.

Flags: Bestimmt, wie der Parameter *Position* interpretiert wird und enthält Informationen über die Veränderungen, die an der Menüoption vorgenommen werden sollen. Er besteht aus einem oder mehreren Werten (siehe folgenden »Hinweis«).

IDNewItem: Bestimmt entweder die Befehls-ID der modifizierten Menüoption oder, wenn *Flags* auf *MF_POPUP* gesetzt ist, das Menü-Handle des Pop-up-Menüs.

NewItem: Bestimmt den Inhalt der geänderten Menüoption. Wenn *Flags* auf *MF_STRING* gesetzt ist (Voreinstellung), ist *NewItem* ein Zeiger auf einen nullterminierten String. Ist *Flags* stattdessen auf *MF_BITMAP* gesetzt, enthält *NewItem* in seinem niederwertigen WORD ein Bitmap-Handle (*HBitmap*).

Wenn *Flags* auf *MF_OWNERDRAW* gesetzt ist, bestimmt *NewItem* einen von der Anwendung bereitgestellten 32-Bit-Wert, den die Anwendung verwenden kann, um zusätzliche Daten zu verwalten, die mit der Menüoption verbunden sind. Dieser 32-Bit-Wert ist für die Anwendung im Feld *itemData* der Struktur verfügbar, auf die der Parameter *lParam* der Botschaften *WM_MEASUREITEM* und *WM_DRAWITEM* zeigt. Diese Botschaften werden übergeben, wenn die Menüoption erstmals angezeigt oder verändert wird.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Ändert sich ein Menü (unabhängig davon, ob es sich in einem sichtbaren Fenster befindet), sollte die Anwendung *DrawMenuBar* aufrufen. Die Veränderung der Attribute vorhandener Menüoptionen läßt sich am schnellsten mit den Funktionen *CheckMenuItem* und *EnableMenuItem* bewerkstelligen.

Jede der folgenden Gruppen enthält Flags, die nicht zusammen benutzt werden sollten:

- *MF_BYCOMMAND , MF_BYPOSITION*
- *MF_DISABLED, MF_ENABLED, MF_GRAYED*
- *MF_BITMAP, MF_STRING, MF_OWNERDISPLAY , MF_SEPARATOR*
- *MF_MENUBARBREAK , MF_MENUBREAK*
- *MF_CHECKED, MF_UNCHECKED*

Die folgende Liste beschreibt die Flags, die im Parameter *Flags* gesetzt werden können:

Wert	Bedeutung
MF_BITMAP	Benutzt ein Bitmap als Option. Das niederwertige WORD des Parameters *NewItem* enthält das Handle des Bitmaps.
MF_BYCOMMAND	Bestimmt, daß der Parameter *Position* die Nummer der Steuerelement-ID der Menüoption angibt (Voreinstellung).

Wert	Bedeutung
MF_BYPOSITION	Bestimmt, daß der Parameter *Position* statt einer ID-Nummer eine zu verändernde Menüoption bezeichnet.
MF_CHECKED	Plaziert eine Auswahlmarkierung neben der Menüoption. Stellt die Anwendung Auswahlmarkierungs-Bitmaps zur Verfügung (siehe auch Funktion *SetMenuItemBitmaps*), bewirkt das Setzen dieses Flags die Darstellung des Bitmaps neben der Menüoption.
MF_DISABLED	Sperrt die Menüoption, so daß sie nicht ausgewählt werden kann, versetzt sie jedoch nicht in den nichtaktivierbaren Zustand.
MF_ENABLED	Aktiviert die Menüoption (sie kann ausgewählt werden) und hebt die Nichtaktivierbarkeit auf.
MF_GRAYED	Sperrt die Menüoption, so daß sie nicht ausgewählt werden kann, und versetzt sie in den nichtaktivierbaren Zustand.
MF_MENUBARBREAK	Gleichbedeutend mit *MF_MENUBREAK* mit der Ausnahme, daß bei Pop-up-Menüs die neue Spalte von der alten durch eine vertikale Linie getrennt wird.
MF_MENUBREAK	Plaziert die Menüoption für statische Menüleistenelemente in einer neuen Zeile. Bei Pop-up-Menüs wird die Menüoption in einer neuen Spalte ohne Trennlinie zwischen den Spalten plaiert.
MF_OWNERDRAW	Bestimmt, daß die Option vom Besitzer gezeichnet wird. Das Fenster, dem das Menü gehört, empfängt eine Botschaft *WM_MEASUREITEM,* wenn das Menü zum ersten Mal dargestellt wird, um Höhe und Breite der Menüoption abzurufen. Die Botschaft *WM_DRAWITEM* wird immer dann an die Anwendung gesendet, wenn die Anwendung das Erscheinungsbild der Menüoption aktualisieren muß. Diese Option ist für Hauptmenüoptionen nicht erlaubt.
MF_POPUP	Bestimmt, daß der Menüoption ein Pop-up-Menü zugeordnet ist. Der Parameter *IDNewItem* bezeichnet ein Handle auf ein Pop-up-Menü, das der Option zugeordnet werden soll. Man sollte das Flag *MF_OWNERDRAW* benutzen, um ein Haupt-Pop-up-Menü oder ein hierarchisches Pop-up-Menü an eine Pop-up-Menüoption anzufügen.

Wert	Bedeutung
MF_SEPARATOR	Zeichnet eine horizontale Trennlinie. Dieses Flag kann in Pop-up-Menüs verwendet werden. Die Linie kann nicht gesperrt, hervorgehoben oder in nichtaktivierbaren Zustand versetzt werden. Windows ignoriert die Parameter *NewItem* und *IDNewItem*.
MF_STRING	Bestimmt, daß die Menüoption ein String ist. Der Parameter *NewItem* zeigt auf den String für die Menüoption.
MF_UNCHECKED	Plaziert keine Auswahlmarkierung neben der Option (Voreinstellung). Wenn die Anwendung Auswahlmarkierungs-Bitmaps zur Verfügung stellt (siehe auch *SetMenuItemBitmaps*), bewirkt das Setzen dieses Flags die Darstellung des Bitmaps neben der Menüoption.

Querverweis *CheckMenuItem, DrawMenuBar, EnableMenuItem, SetMenuItemBitmaps*

MoveTo

Syntax
```
DWORD MoveTo(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion stellt den Punkt, der durch die Parameter *X* und *Y* bestimmt wird, als aktuelle Position ein.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate der neuen Position.

Y: Bestimmt die virtuelle y-Koordinate der neuen Position.

Rückgabewert Es werden die x- und y-Koordinate der vorherigen Position zurückgeliefert. Die y-Koordinate liegt im höherwertigen, die x-Koordinate im niederwertigen WORD.

Hinweis Obwohl *MoveTo* keine Ausgabe bewirkt, beeinflußt sie andere Ausgabefunktionen, die die aktuelle Position verwenden.

MoveWindow

Syntax
```
void MoveWindow(HWND Wnd, int X, int Y, int Width,
int Height, BOOL Repaint);
```

Beschreibung Diese Funktion bewirkt, daß eine Botschaft *WM_SIZE* an das vorgegebene Fenster gesendet wird. Die Parameter *X*, *Y*, *Width* und *Height* geben die neue Größe des Fensters vor.

Wnd: Bezeichnet ein untergeordnetes oder Pop-up-Fenster.

X: Bestimmt die neue x-Koordinate der oberen linken Ecke des Fensters.

Y: Bestimmt die neue y-Koordinate der oberen linken Ecke des Fensters. Für Pop-up-Fenster sind *X* und *Y* in Bildschirmkoordinaten gegeben (relativ zur oberen linken Ecke des Bildschirms). Für untergeordnete Fenster gelten sie in Client-Koordinaten (relativ zur oberen linken Ecke des Client-Bereichs des übergeordneten Fensters).

Width: Bestimmt die neue Breite des Fensters.

Height: Bestimmt die neue Höhe des Fensters.

Repaint: Bestimmt, ob das Fenster nach der Verschiebung neu gezeichnet werden soll. Wenn *Repaint* 0 ist, wird das Fenster nicht neu gezeichnet.

Rückgabewert Keiner.

Hinweis Jedes untergeordnete oder Pop-up-Fenster hat eine minimale Höhe und Breite. Diese Werte hängen von Art und Inhalt des Fensters ab. Jeder Versuch, die Höhe und Breite mit der Funktion *MoveWindow* weiter zu verkleinern, schlägt fehl. Die Botschaft *WM_SIZE*, die von dieser Funktion erzeugt wird, gibt die neue Breite und Höhe des Client-Bereichs des Fensters weiter, nicht die des gesamten Fensters.

MulDiv

Syntax `int MulDiv(int Number, int Numerator, int Denominator);`

Beschreibung Diese Funktion multipliziert zwei Integerwerte und dividiert das Ergebnis durch einen dritten (was dem gerundeten Produkt eines Integers und einer rationalen Zahl entspricht). Der Rückgabewert ist das Endergebnis, gerundet auf den nächstliegenden Integer.

Number: Bestimmt die Zahl, die mit *Numerator* multipliziert werden soll.

Numerator: Bestimmt die Zahl, die mit *Number* multipliziert werden soll (Zähler).

Denominator: Bestimmt die Zahl, durch die das Resultat der Multiplikation dividiert werden soll (Nenner).

Rückgabewert Bei erfolgreicher Ausführung wird das Ergebnis der Multiplikation und der Division zurückgeliefert. Wenn entweder ein Überlauf auftritt oder *Denominator* auf 0 gesetzt war, ist der Rückgabewert 32767 oder -32767.

OemKeyScan

Syntax
```
DWORD OemKeyScan(WORD OemChar);
```

Beschreibung Diese Funktion überträgt die OEM-ASCII-Codes 0x000 bis 0x0FF in die OEM Scan-Codes und Shift-Zustände. Sie stellt dadurch Informationen bereit, die es einem Programm ermöglichen, einen OEM-Text an ein anderes Programm als simulierte Tastatureingabe zu übergeben. Speziell hierfür wird sie von Windows insbesondere im erweiterten 386-Modus verwendet.

OemChar: Bestimmt den ASCII-Wert des OEM-Zeichens.

Rückgabewert Der Rückgabewert enthält in seinem niederwertigen WORD den Scan-Code des OEM-Zeichens, das durch den Parameter *OemChar* bezeichnet wird. Das höherwertige WORD des Rückgabewerts enthält Flags, die den Shift-Zustand angeben. Die einzelnen Flags haben folgende Bedeutung:

Bit	Bedeutung
2	Ctrl-Taste ist gedrückt.
1	Eine Shift-Taste ist gedrückt.

Wenn das Zeichen nicht in der OEM-Zeichentabelle definiert ist, enthält sowohl das höherwertige als auch das niederwertige WORD des Rückgabewerts -1.

Hinweis Diese Funktion sieht keine Übersetzungen für Zeichen vor, die Ctrl-, Alt- oder Sondertasten benötigen. Zeichen, die von ihr nicht übersetzt werden, müssen durch Simulation der Eingabe mittels des ALT+Nummernblock-Mechanismus kopiert werden. Die Taste Numlock darf dabei nicht arretiert sein.

Diese Funktion ruft in neueren Versionen des Tastaturtreibers die Funktion *VkKeyScan* auf.

OemToAnsi

Syntax
```
BOOL OemToAnsi(LPSTR OemStr, LPSTR AnsiStr);
```

Beschreibung Diese Funktion übersetzt den String, auf den der Parameter *OemStr* zeigt, vom OEM-Zeichensatz in den ANSI-Zeichensatz. Der String kann größer als 64 KByte sein.

OemStr: Zeigt auf einen nullterminierten String aus dem OEM-Zeichensatz.

AnsiStr: Zeigt auf die Position, an die der übersetzte String kopiert werden soll. *AnsiStr* kann mit *OemStr* identisch sein, um den String an der aktuellen Position zu übersetzen.

Rückgabewert Der Rückgabewert ist immer -1.

OemToAnsiBuff

Syntax
```
void OemToAnsiBuff(LPSTR OemStr, LPSTR AnsiStr,
int Length);
```

Beschreibung Diese Funktion übersetzt den String in dem Puffer, auf den der Parameter *OemStr* zeigt, vom OEM- in den ANSI-Zeichensatz.

OemStr: Zeigt auf einen Puffer, der ein oder mehrere Zeichen aus dem OEM-Zeichensatz enthält.

AnsiStr: Zeigt auf die Position, an die der String kopiert werden soll. Der Parameter *AnsiStr* kann mit *OemStr* identisch sein, um den String an der aktuellen Position zu übersetzen.

Length: Bestimmt die Zahl der Zeichen in dem durch *OemStr* bezeichneten Puffer. Ist *Length* 0, ist die Länge 64 KByte (65536).

Rückgabewert Keiner.

OffsetClipRgn

Syntax
```
int OffsetClipRgn(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion verschiebt die Clipping-Region des gegebenen Gerätes um die vorgegebenen Offsetwerte. (*X* Einheiten entlang der x-Achse und *Y* Einheiten entlang der y-Achse).

DC: Bezeichnet den Gerätekontext.

X: Bezeichnet die Anzahl der virtuellen Einheiten, um die nach links oder rechts verschoben werden soll.

Y: Bezeichnet die Zahl der virtuellen Einheiten, um die nach oben oder unten verschoben werden soll.

Rückgabewert Der Rückgabewert bezeichnet den Typ der neuen Region. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
COMPLEXREGION	Clipping-Region hat überlappende Grenzen.
ERROR	Gerätekontext ist ungültig.
NULLREGION	Clipping-Region ist leer.
SIMPLEREGION	Clipping-Region hat keine überlappenden Grenzen.

OffsetRect

Syntax
```
void OffsetRect(LPRECT Rect, int X, int Y);
```

Beschreibung Diese Funktion verschiebt das bezeichnete Rechteck um *X* Einheiten entlang der x-Achse und um *Y* Einheiten entlang der

Y-Achse. Die Parameter *X* und *Y* sind vorzeichenbehaftet, so daß das Rechteck nach rechts oder links und nach oben oder unten bewegt werden kann.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die das zu verschiebende Rechteck enthält.

X: Bestimmt den Betrag der Verschiebung nach rechts oder links. Zum Verschieben nach links muß er negativ sein.

Y: Bestimmt den Betrag der Verschiebung nach oben oder unten. Zum Verschieben nach oben muß er negativ sein.

Rückgabewert Keiner.

Hinweis Die Koordinatenwerte eines Rechtecks dürfen nicht größer als 32767 oder kleiner als -32768 sein.

OffsetRgn

Syntax
```
int OffsetRgn(HRGN Rgn, int X, int Y);
```

Beschreibung Diese Funktion verschiebt die angegebene Region um die gegebenen Offsetwerte und zwar um *X* Einheiten entlang der x-Achse und um *Y* Einheiten entlang der y-Achse.

Rgn: Bezeichnet die zu verschiebenden Region.

X: Bezeichnet die Anzahl der Einheiten, um die nach links oder rechts verschoben werden soll.

Y: Bezeichnet die Anzahl der Einheiten, um die nach oben oder unten verschoben werden soll.

Rückgabewert Der Rückgabewert bezeichnet den Typ der neuen Region. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
COMPLEXREGION	Region hat überlappende Grenzen.
ERROR	Handle der Region ist ungültig.
NULLREGION	Region ist leer.
SIMPLEREGION	Region hat keine überlappenden Grenzen.

Hinweis Die Koordinatenwerte der Region dürfen nicht größer als 32767 und nicht kleiner als -32768 sein.

OffsetViewportOrg

Syntax
```
DWORD OffsetViewportOrg(HDC DC, int X, int Y);
```

Beschreibung Die Funktion verschiebt den Ursprung des Grafikfensters (VO) relativ zu den aktuellen Koordinaten nach folgender Formel:

$$xNeuVO = xAltVO + X;\ yNeuVO = yAltVO + Y$$

Der neue Ursprung ist die Summe des aktuellen Ursprungs und der Werte *X* und *Y*.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die Anzahl der gerätespezifischen Einheiten, die zur x-Koordinate des aktuellen Ursprungs addiert werden sollen.

Y: Bestimmt die Anzahl der gerätespezifischen Einheiten, die zur y-Koordinate des aktuellen Ursprungs addiert werden sollen.

Rückgabewert Der Rückgabewert ist der vorherige Ursprung des Fensters in gerätespezifischen Koordinaten. Die vorherige y-Koordinate liegt im höherwertigen, die vorherige x-Koordinate im niederwertigen WORD.

OffsetWindowOrg

Syntax
```
DWORD OffsetWindowOrg(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion verändert den Ursprung des Fensters (WO) relativ zu den aktuellen Werten gemäß der folgenden Formel:

$$xNeuWO = xAltWO + X;\ yNeuWO = yAltWO + Y$$

Der neue Ursprung ist die Summe des aktuellen Ursprungs und der Werte *X* und *Y*.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die Anzahl der virtuellen Einheiten, die zur x-Koordinate des aktuellen Ursprungs addiert werden sollen.

Y: Bezeichnet die Anzahl der virtuellen Einheiten, die zur y-Koordinate des aktuellen Ursprungs addiert werden sollen.

Rückgabewert Der Rückgabewert ist der vorherige Fensterursprung in virtuellen Koordinaten. Die vorherige y-Koordinate steht im höherwertigen, die vorherige x-Koordinate im niederwertigen WORD.

OpenClipboard

Syntax
```
BOOL OpenClipboard(HWND Wnd);
```

Beschreibung Diese Funktion öffnet die Zwischenablage für exklusiven Zugriff und hindert so andere Anwendungen daran, den Inhalt der Zwischenablage zu ändern.

Wnd: Bezeichnet das Fenster, welches mit der geöffneten Zwischenablage verbunden werden soll.

Rückgabewert Der Rückgabewert bezeichnet den Status der Zwischenablage. Er ist ungleich *Null*, wenn die Zwischenablage geöffnet ist und *Null*, wenn die Zwischenablage schon von einer anderen Anwendung geöffnet wurde.

Hinweis Eine Anwendung sollte die Funktion *CloseClipboard* nach jedem erfolgreichen Aufruf von *OpenClipboard* verwenden.

Querverweis *CloseClipboard*

OpenComm

Syntax
```
int OpenComm(LPSTR ComName, WORD InQueue, WORD
OutQueue);
```

Beschreibung Diese Funktion öffnet eine Schnittstelle und weist ihr ein Handle zu. Die Schnittstelle wird für eine Standardkonfiguration initialisiert. Um sie mit anderen Werten zu initialisieren, sollte die Funktion *SetCommState* verwendet werden. Die Funktion *OpenComm* belegt Speicherplatz für Empfangs- und Sende-Warteschlangen. Die Warteschlangen werden von der interruptgesteuerten Sende-/Empfangssoftware benutzt.

ComName: Weist auf einen String, der COM*n* oder LPT*n* enthält, wobei *n* im Bereich von 1 bis zur Anzahl der verfügbaren Schnittstellen für den jeweiligen I/O-Anschluß liegt.

InQueue: Bestimmt die Größe der Empfangs-Warteschlange.

OutQueue: Bestimmt die Größe der Sende-Warteschlange.

Rückgabewert Der Rückgabewert bezeichnet die geöffnete Schnittstelle. Im Fall eines Fehlers wird einer der folgenden negativen Fehlerwerte zurückgeliefert:

Wert	Bedeutung
IE_BADID	Ungültige oder nicht unterstützte ID.
IE_BAUDRATE	Nicht unterstützte Baudrate.
IE_BYTESIZE	Ungültige Bytegröße.
IE_DEFAULT	Fehler in Standardparametern.
IE_HARDWARE	Hardware nicht vorhanden.
IE_MEMORY	Es können keine Warteschlangen eingerichtet werden.
IE_NOPEN	Gerät nicht bereit.
IE_OPEN	Schnittstelle bereits geöffnet.

Hinweis LPT-Anschlüsse sind nicht interruptgesteuert – für diese werden die Parameter *InQueue* und *OutQueue* ignoriert und die Größe der Warteschlange auf 0 gesetzt.

Querverweis *CloseComm*, *GetCommError*, *GetCommEventMask*, *GetCommState*, *SetCommBreak*, *SetCommEventMask*, *SetCommState*

OpenFile

Syntax
```
int OpenFile(LPSTR FileName, LPOFSTRUCT ReOpenBuff,
WORD Style);
```

Beschreibung Diese Funktion führt eine der folgenden Datei-Operationen aus: Anlegen, Öffnen, Wiederöffnen, Löschen).

FileName: Zeigt auf einen nullterminierten String, der die betroffene Datei benennt. Der String muß aus Zeichen aus dem ANSI-Zeichensatz bestehen.

ReOpenBuff: Zeigt auf eine *OFSTRUCT*-Datenstruktur, die dazu bestimmt ist, Informationen über die Datei aufzunehmen, wenn diese erstmals geöffnet wird. Die Struktur kann bei späteren Aufrufen von *OpenFile* dazu verwendet werden, sich auf die geöffnete Datei zu beziehen. Das Feld *szPathName* dieser Struktur enthält Zeichen aus dem OEM- Zeichensatz.

Style: Bestimmt die auszuführende Operation. Folgende Werte können mit dem bitweisen ODER-Operator (|) kombiniert werden.

Wert	Bedeutung
OF_CANCEL	Fügt einen Aktionsschalter für *Abbruch* in das Dialogfenster *OF_PROMPT* ein. Die Betätigung des Abbruchschalters weist *OpenFile* an, eine Fehlermeldung `Datei nicht gefunden` zurückzugeben.
OF_CREATE	Weist *OpenFile* an, eine neue Datei anzulegen. Falls die Datei schon vorhanden ist, wird sie auf die Länge 0 gekürzt.
OF_DELETE	Löscht die Datei.
OF_EXIST	Öffnet die Datei und schließt sie wieder. Der Rückgabewert wird für eine Prüfung auf die Existenz einer Datei verwendet.
OF_PARSE	Füllt die Datenstruktur *OFSTRUCT*, führt aber keine weitere Aktionen aus.
OF_PROMPT	Zeigt ein Dialogfenster an, wenn die angesprochene Datei nicht existiert. Das Dialogfenster informiert den Benutzer, daß Windows die Datei nicht finden kann und fordert den Benutzer auf, eine Diskette mit der Datei in Laufwerk A einzulegen.

Wert	Bedeutung
OF_READ	Öffnet die Datei nur zum Lesen.
OF_READWRITE	Öffnet die Datei zum Lesen und Schreiben.
OF_REOPEN	Öffnet die Datei unter Verwendung der Informationen im Datenpuffer für erneutes Öffnen.
OF_SHARE_COMPACT	Öffnet eine Datei im Kompatibilitätsmodus und ermöglicht dadurch das Öffnen der Datei, sooft es nötig ist. *OpenFile* kann nicht ausgeführt werden, wenn die Datei von einem anderen Prozeß im Kompatibilitätsmodus geöffnet wurde.
OF_SHARE_DENY_NONE	Öffnet die Datei, ohne anderen Prozessen den Lese- oder Schreibzugriff auf diese Datei zu verweigern. *OpenFile* kann nicht ausgeführt werden, wenn die Datei von einem anderen Prozeß im Kompatibilitätsmodus geöffnet wurde.
OF_SHARE_DENY_READ	Öffnet die Datei und verweigert anderen Prozessen den Lesezugriff auf die Datei. *OpenFile* kann nicht ausgeführt werden, wenn die Datei im Kompatibilitätsmodus oder für Lesezugriff von einem anderen Prozeß geöffnet wurde.
OF_SHARE_DENY_WRITE	Öffnet die Datei und verweigert anderen Prozessen Schreibzugriff auf die Datei. *OpenFile* kann nicht ausgeführt werden, wenn die Datei im Kompatibilitätsmodus oder für Schreibzugriff von einem anderen Prozeß geöffnet wurde.
OF_SHARE_EXCLUSIVE	Öffnet die Datei im Exklusivmodus und verweigert anderen Prozessen sowohl Lese- als auch Schreibzugriff auf die Datei. *OpenFile* kann nicht ausgeführt werden, wenn die Datei in irgendeinem anderen Modus für Lese- oder Schreibzugriff geöffnet wurde. Letzteres gilt auch für den aktuellen Prozeß.
OF_VERIFY	Überprüft, ob Datum und Zeit der Datei dieselben sind, wie beim vorherigen Öffnen. Dieser Wert ist zur zusätzlichen Überprüfung von schreibgeschützten Dateien nützlich.
OF_WRITE	Öffnet die Datei nur zum Schreiben.

Rückgabewert Bei erfolgreicher Ausführung wird ein DOS-Datei-Handle zurückgeliefert, andernfalls -1.

Hinweis Wenn der Parameter *FileName* nur einen Dateinamen und die Namenserweiterung angibt, sucht die Funktion eine entsprechende Datei in den folgenden Verzeichnissen:

1. Im aktuellen Verzeichnis.

2. Im Windows-Verzeichnis (dem Verzeichnis, das WIN.COM enthält); die Funktion *GetWindowsDirectory* ermittelt den Pfadnamen dieses Verzeichnisses.

3. Im Windows-Systemverzeichnis (dem Verzeichnis, das Systemdateien wie KERNEL.EXE enthält); die Funktion *GetSystemDirectory* ermittelt den Pfadnamen dieses Verzeichnisses.

4. In den Verzeichnissen, die in der Environment-Variablen PATH aufgelistet sind.

5. In den Verzeichnissen, die in einem Netz im Befehl MAP angegeben wurden.

Windows durchsucht die Verzeichnisse in der angegebenen Reihenfolge. *FileName* darf keine Platzhalterzeichen enthalten.

Um die Datei nach dem Gebrauch zu schließen, sollte die Anwendung die Funktion *_lclose* aufrufen.

OpenIcon

Syntax
```
BOOL OpenIcon(HWND Wnd);
```

Beschreibung Diese Funktion aktiviert ein zum Symbol verkleinertes Fenster und zeigt es an. Windows stellt es in ursprünglicher Größe und an der ursprünglichen Position wieder her.

Wnd: Bezeichnet das Fenster.

Rückgabewert Bei erfolgreicher Ausführung der Funktion wird ungleich Null zurückgeliefert, andernfalls Null.

OpenSound

Syntax
```
int OpenSound(void);
```

Beschreibung Diese Funktion greift auf das Klangausgabegerät zu und verhindert, daß es nachfolgend von anderen Anwendungen angesteuert wird.

Die Funktion hat keine Parameter.

Rückgabewert Bei erfolgreicher Ausführung wird die Anzahl der verfügbaren Töne zurückgeliefert. Wenn das Klangausgabegerät bereits ver-

wendet wird, ist der Rückgabewert *S_SERDVNA*. Ist nicht ausreichend Speicherplatz verfügbar, ist er *S_SEROFM*.

OutputDebugString

Syntax
```
void OutputDebugString(LPSTR OutputString);
```

Beschreibung Diese Funktion übergibt eine Debuggerbotschaft an den Debugger, wenn dieser vorhanden ist, andernfalls an die serielle Standardschnittstelle (AUX).

OutputString: Zeigt auf einen nullterminierten String.

Rückgabewert Keiner.

Hinweis Diese Funktion sichert alle Register. Sie ist nur in der Debugger-Version von Windows verfügbar.

PaintRgn

Syntax
```
BOOL PaintRgn(HDC DC, HRGN Rgn);
```

Beschreibung Diese Funktion füllt die vom Parameter *Rgn* bezeichnete Region mit dem ausgewählten Pinsel aus.

DC: Bezeichnet den Gerätekontext, der die Region enthält.

Rgn: Bezeichnet die Region, die ausgefüllt werden soll. Die Koordinaten für die angegebene Region sind in gerätespezifischen Einheiten angegeben.

Rückgabewert Bei erfolgreicher Ausführung wird ungleich Null zurückgeliefert, andernfalls Null.

PaletteIndex

Syntax
```
COLORREF PaletteIndex(int PaletteIndex);
```

Beschreibung Dieses Makro nimmt den Index eines virtuellen Paletteneintrags entgegen und gibt einen Wert zurück, der aus einer 1 im höherwertigen Byte und dem Paletteneintragsindex im niederwertigen Byte besteht. Dieser wird Paletteneintragsbezeichner genannt.

Eine Anwendung, die eine Farbpalette realisiert, kann diesen Bezeichner für Funktionen, die eine Farbe erwarten, anstelle eines expliziten RGB-Wertes verwenden. Dies erlaubt der Funktion, die Farbe im angegebenen Paletteneintrag zu verwenden.

PaletteIndex: Bestimmt einen Index zu dem Paletteneintrag, der die Farbe enthält, welche für eine Grafikoperation benutzt werden soll.

Rückgabewert Der Rückgabewert ist ein Indexbezeichner zu einer virtuellen Palette. Bei Verwendung einer virtuellen Palette kann eine Anwendung anstelle eines expliziten RGB-Wertes diesen Bezeichner für GDI-Funktionen einsetzen, die eine Farbe anfordern.

PaletteRGB

Syntax
```
DWORD PaletteRGB(BYTE Red, BYTE Green, BYTE Blue);
```

Beschreibung Dieses Makro nimmt drei Werte entgegen, die die relativen Intensitäten von Rot, Grün und Blau darstellen, und gibt einen Wert zurück, der im höherwertigen Byte aus einer 2 und in den drei niederwertigen Bytes aus einem RGB-Wert besteht. Dieser Wert wird auch palettenbezogener RGB-Bezeichner genannt. Bei der Verwendung einer Farbpalette kann eine Anwendung diesen Bezeichner anstelle eines expliziten RGB-Werts an Funktionen übergeben, die eine Farbe anfordern.

Für Ausgabegeräte, die virtuelle Paletten unterstützen, gleicht Windows einen palettenbezogenen RGB-Wert an den nächsten Farbwert in der virtuellen Palette des Gerätekontexts an, als ob die Anwendung einen Index zu diesem Paletteneintrag angegeben hätte.

Wenn ein Ausgabegerät keine Systempalette unterstützt, verwendet Windows den palettenbezogenen RGB-Wert, als wäre er ein von der Funktion *RGB* zurückgegebener konventioneller *RGB*-Wert.

Red: Bestimmt die Intensität des roten Farbwerts (0x000 ist minimale, 0x0FF ist maximale Intensität.

Green: Bestimmt die Intensität des grünen Farbwerts.

Blue: Bestimmt die Intensität des blauen Farbwerts.

Rückgabewert Es wird ein palettenbezogener RGB-Wert zurückgegeben.

PatBlt

Syntax
```
BOOL PatBlt(HDC DC, int X, int Y, int Width, int
Height, DWORD Rop);
```

Beschreibung Diese Funktion erzeugt auf dem angegebenen Gerät ein Bitmuster.

Das Muster ist eine Kombination des ausgewählten Pinsels mit dem Muster, das sich schon auf dem Gerät befindet. Der vom Parameter *Rop* angegebene Rasteroperationscode legt fest, wie die Muster kombiniert werden sollen.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Rechtecks, das das Muster aufnehmen soll.

Y: Bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Rechtecks, das das Muster aufnehmen soll.

Width: Bestimmt die Breite des Rechtecks, das das Muster aufnehmen soll, in virtuellen Einheiten.

Height: Bestimmt die Höhe des Rechtecks, das das Muster aufnehmen soll, in virtuellen Einheiten.

Rop: Der Parameter *Rop* bestimmt den Rasteroperationscode. Möglich sind die folgenden Werte:

Code	Beschreibung
PATCOPY	Kopiert das vorgegebene Muster in das Ziel-Bitmap.
PATINVERT	Kombiniert Ziel-Bitmap und Muster unter Verwendung des logischen-Exclusives-ODER-Operators.
DSTINVERT	Invertiert das Ziel-Bitmap.
BLACKNESS	Erzeugt eine völlig geschwärzte Ausgabe.
WHITENESS	Erzeugt eine völlig weiße Ausgabe.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Bitmuster gezeichnet wurde, andernfalls ist er Null.

Hinweis Die Werte von *Rop* für diese Funktion sind eine begrenzte Untermenge der insgesamt 256 ternären Rasteroperationscodes; insbesondere kann ein Operationscode, der sich auf eine Quelle bezieht, nicht verwendet werden.

Nicht alle Geräte unterstützen die Funktion *PatBlt*. Weitere Informationen finden sich in der Beschreibung der Funktion *GetDeviceCaps*.

PeekMessage

Syntax
```
BOOL PeekMessage(LPMSG Msg, HWND Wnd,
WORD MsgFilterMin, WORD MsgFilterMax, WORD RemoveMsg);
```

Beschreibung Diese Funktion durchsucht die Anwendungswarteschlange nach einer Botschaft und legt die Botschaft (falls vorhanden) in der *MSG*-Struktur, auf die *Msg* zeigt, ab. Im Gegensatz zur Funktion *GetMessage* wartet *PeekMessage* nicht darauf, daß die Botschaft in die Warteschlange gestellt wird, bevor sie zurückkehrt. Sie übergibt jedoch die Steuerung (wenn das Flag *PM_NOYIELD* nicht gesetzt ist) und behält sie nach dieser Übergabe solange, bis Windows sie an die Anwendung zurückgibt. *PeekMessage* liefert innerhalb des durch die Parameter

MsgFilterMin und *MsgFilterMax* bestimmten Bereichs von Botschaftswerten nur Botschaften, die dem durch den Parameter *Wnd* bezeichneten Fenster oder dessen, von der Funktion *IsChild* ermittelten, untergeordneten Fenstern zugeordnet sind.

Wenn *Wnd* 0 ist, liefert *PeekMessage* Botschaften für jedes Fenster, das zur aufrufenden Anwendung gehört. *PeekMessage* liefert keine Botschaften für Fenster, die zu anderen Anwendungen gehören. Wenn *Wnd* -1 ist, gibt *PeekMessage* nur die Botschaften mit einem *Wnd* von 0 zurück, die von der Funktion *PostAppMessage* übermittelt werden. Wenn sowohl *MsgFilterMin* als auch *MsgFilterMax* 0 sind, gibt *PeekMessage* alle verfügbaren Botschaften zurück (es wird keine Bereichsfilterung durchgeführt).

Die Flags *WM_KEYFIRST* und *WM_KEYLAST* können als Filterwerte zum Liefern aller Tastaturbotschaften benutzt werden; die Flags *WM_MOUSEFIRST* und *WM_MOUSELAST* können zum Liefern aller Mausbotschaften benutzt werden.

Msg: Zeigt auf eine *MSG*-Datenstruktur, welche die Botschaftsinformationen aus der Windows-Anwendungswarteschlange enthält.

Wnd: Bezeichnet das Fenster, dessen Botschaften überprüft werden sollen.

MsgFilterMin: Bestimmt den Wert der untersten Botschafts-ID, die überprüft werden soll.

MsgFilterMax: Bestimmt den Wert der höchsten Botschafts-ID, die überprüft werden soll.

RemoveMsg: Bestimmt eine Kombination der Flags, die nachfolgend beschrieben sind. *PM_NOYIELD* kann entweder mit *PM_NOREMOVE* oder mit *PM_REMOVE* kombiniert werden.

Wert	Bedeutung
PM_NOREMOVE	Botschaften werden nach der Bearbeitung durch *PeekMessage* nicht aus der Botschaftswarteschlange gelöscht.
PM_NOYIELD	Hindert die aktuelle Task daran, zu unterbrechen und Systemressourcen an eine andere Task abzugeben.
PM_REMOVE	Botschaften werden nach der Bearbeitung durch *PeekMessage* aus der Warteschlange gelöscht.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn eine Botschaft verfügbar ist, andernfalls ist er Null.

Hinweis

PeekMessage löscht keine *WM_PAINT*-Botschaften aus der Warteschlange, diese verbleiben dort bis zu ihrer Bearbeitung. Jede Anwendung sollte aufrufen *GetMessage*, *PeekMessage* und *WaitMessage*, da dies die einzige Möglichkeit ist, die Steuerung an andere Anwendungen zu übergeben.

Wenn *GetMessage*, *PeekMessage* und *WaitMessage* die Steuerung an andere Anwendungen abgeben, können sich die Stack- und Datensegmente der Anwendung, die die Funktion aufruft, im Arbeitsspeicher verschieben, um sich an den veränderten Speicherbedarf anderer Anwendungen anzupassen.

Wenn die Anwendung Zeiger auf Objekte im Daten- oder Stack-Segment gespeichert hat (globale oder lokale Variablen) und deren Sperren aufgehoben sind, können diese durch einen Aufruf von *GetMessage*, *PeekMessage* oder *WaitMessage* ungültig werden. Der Parameter *Msg* der aufgerufenen Funktion bleibt in jedem Fall gültig.

Querverweis

GetMessage, *WaitMessage*

Pie

Syntax

```
BOOL Pie(HDC DC, int X1, int Y1, int X2, int Y2,
int X3, int Y3, int X4, int Y4);
```

Beschreibung

Diese Funktion zeichnet ein Tortendiagramm, indem sie einen Ellipsenbogen angibt, dessen Mittelpunkt (der Mittelpunkt der ganzen Ellipse) und zwei Endpunkte durch Linien miteinander verbunden werden.

Der Mittelpunkt des Bogens ist der Mittelpunkt des umgrenzenden Rechtecks, das die Parameter *X1*, *Y1*, *X2* und *Y2* vorgeben. Start- und Endpunkte des Bogens sind die Schnittpunkte der Ellipse mit der Linie von (*X3*, *Y3*) nach (*X4*, *Y4*). Der Bogen wird mit dem ausgewählten Stift entgegen dem Uhrzeigersinn gezeichnet, und zusätzlich werden die beiden Endpunkte des Bogens mit seinem Mittelpunkt durch zwei Linien verbunden.

Das Tortenstück wird mit dem ausgewählten Pinsel ausgefüllt.

Wenn *X3* gleich *X4* und *Y3* gleich *Y4* ist, dann ist das Ergebnis eine Ellipse mit einer einzigen Linie vom Mittelpunkt der Ellipse zum Schnittpunkt der Linie vom Mittelpunkt nach (*X3*,*Y3*) mit der Ellipse.

DC: Bezeichnet den Gerätekontext.

X1, Y1: Bestimmt die obere linke Ecke des umgebenden Rechtecks in virtuellen Koordinaten.

X2, Y2: Bestimmt die untere rechte Ecke des umgebenden Rechtecks in virtuellen Koordinaten.

X3, Y3: Bestimmt den einen Endpunkt der Schnittlinie in virtuellen Koordinaten. Der Endpunkt muß nicht auf der Ellipse liegen.

X4, Y4: Bestimmt den anderen Endpunkt der Schnittlinie in virtuellen Koordinaten. Der Endpunkt muß nicht auf der Ellipse liegen.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Tortenstück gezeichnet wurde, andernfalls ist er Null.

Hinweis Die Breite des Rechtecks, die durch die Differenz *X2-X1* gegeben ist, darf 32767 Einheiten nicht überschreiten. Diese Einschränkung gilt auch für die Höhe des Rechtecks (*Y2-Y1*).

Die aktuelle Position wird von dieser Funktion weder verwendet noch aktualisiert.

PlayMetaFile

Syntax
```
BOOL PlayMetaFile(HDC DC, HANDLE MF);
```

Beschreibung Diese Funktion spielt den Inhalt des bezeichneten Metafiles auf dem angegebenen Gerät ab. Das Metafile kann beliebig oft wiedergegeben werden.

DC: Bezeichnet den Gerätekontext des Ausgabegerätes.

MF: Bezeichnet das Metafile.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich ist, andernfalls ist er Null.

PlayMetaFileRecord

Syntax
```
void PlayMetaFileRecord(HDC DC,
LPHANDLETABLE HandleTable, LPMETARECORD MetaRecord,
WORD Handles);
```

Beschreibung Diese Funktion spielt ein Metafile-Record durch Ausführung des GDI-Funktionsaufrufes ab, der in dem Metafile-Record enthalten ist.

DC: Bezeichnet den Gerätekontext des Ausgabegeräts.

HandleTable: Zeigt auf eine Objekt-Handle-Tabelle (*HANDLETABLE*-Struktur), die für die Metafile-Wiedergabe verwendet werden soll.

MetaRecord: Zeigt auf die wiederzugebende *METARECORD*-Struktur des Metafile.

Handles: Bestimmt die Anzahl der Handles in der Handle-Tabelle.

Rückgabewert Keiner.

Hinweis Eine Anwendung verwendet diese Funktion üblicherweise in Verbindung mit der Funktion *EnumMetafile*, um ein Metafile erst zu modifizieren und dann wiederzugeben.

Querverweis *EnumMetaFile*

Polygon

Syntax
```
BOOL Polygon(HDC DC, LPPOINT Points, int Count);
```

Beschreibung Diese Funktion verbindet zwei oder mehr Punkte (Eckpunkte) durch Linien, wodurch ein oder mehrerer Polygone entstehen. Die Polygone werden unter Verwendung des aktuellen Polygonfüllmodus ausgefüllt.

Eine Beschreibung der Polygonfüllmodi findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »Polygonfüllmodi«. Die Figur wird, falls dies erforderlich ist, durch Zeichnen einer Linie vom letzten Eckpunkt zum ersten automatisch geschlossen.

DC: Bezeichnet den Gerätekontext.

Points: Zeigt auf ein Array von Punkten, die die Eckpunkte des Polygons bezeichnen. Jeder Punkt im Array ist eine *POINT*-Datenstruktur.

Count: Bestimmt die Anzahl von Eckpunkten, die im Array beschrieben sind.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Die aktuelle Position wird von dieser Funktion weder benutzt noch aktualisiert.

Der aktuelle Polygonfüllmodus kann mit der Funktion *GetPolyFillMode* abgerufen und mit der Funktion *SetPolyFillMode* gesetzt werden.

Querverweis *SetPolyFillMode*

Polyline

Syntax
```
BOOL Polyline(HDC DC, LPPOINT Points, int Count);
```

Beschreibung Diese Funktion zeichnet eine Folge von Linienzügen, die die vom Parameter *Points* bezeichneten Punkte der Reihe nach verbinden. Das Ergebnis ist dasselbe, das durch Verwendung der Funktionen *MoveTo* und *LineTo* erreicht wird. Die aktuelle Position wird von der Funktion *Polyline* weder benutzt noch aktualisiert.

DC: Bezeichnet den Gerätekontext.

Points: Zeigt auf ein Array von Punkten, die verbunden werden sollen. Jeder Punkt in dem Array ist eine *POINT*-Datenstruktur.

Count: Bestimmt die Anzahl der Punkte im Array. Der Parameter *Count* muß mindestens 2 sein.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Liniensegmente gezeichnet wurden, andernfalls ist er Null.

Hinweis *Polyline* zeichnet die Linien mit dem ausgewählten Stift.

PolyPolygon

Syntax
```
BOOL PolyPolygon(HDC DC, LPPOINT Points,
LPPOINT PolyCounts, int Count);
```

Beschreibung Diese Funktion erzeugt eine Reihe von Polygonen. Die Polygone werden unter Verwendung des aktuellen Polygonfüllmodus ausgefüllt. Eine Beschreibung der Polygonfüllmodi findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »Polygonfüllmodi«. Die Polygone dürfen sich überlappen.

DC: Bezeichnet den Gerätekontext.

Points: Zeigt auf ein Array von *POINT*-Datenstrukturen, die der Reihe nach die Eckpunkte mehrerer Polygone definieren. Jedes Polygon muß geschlossen sein. Im Gegensatz zu Polygonen, die mit der Funktion *Polygon* erzeugt werden, werden mit *PolyPolygon* erzeugte Polygone nicht automatisch geschlossen.

PolyCounts: Zeigt auf ein Array von Integerwerten, die für jedes Polygon die jeweilige Anzahl der Punkte im Array *Points* bezeichnet.

Count: Bestimmt die Gesamtzahl von Integerwerten im Array *PolyCounts*.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn die Polygone gezeichnet wurden, andernfalls ist er 0.

PostAppMessage

Syntax
```
BOOL PostAppMessage(HANDLE Task, WORD Msg, WORD wParam,
LONG lParam);
```

Beschreibung Diese Funktion übermittelt eine Botschaft an eine Anwendung, die von einem Task-Handle bezeichnet wird und kehrt dann zurück, ohne auf Bestätigung zu warten. Die Anwendung, welche die Botschaft empfängt, liefert diese, indem sie eine der Funktionen *GetMessage* oder *PeekMessage* aufruft. Der

Parameter *hWnd* der zurückgegebenen *Msg*-Datenstruktur ist NULL.

Task: Bezeichnet die Task, die die Botschaft empfangen soll. Die Funktion *GetCurrentTask* gibt dieses Handle zurück.

Msg: Bestimmt den Typ der übermittelten Botschaft.

wParam: Bestimmt zusätzliche Botschaftsinformationen.

lParam: Bestimmt zusätzliche Botschaftsinformationen.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Botschaft abgesetzt wurde, andernfalls ist er Null.

Querverweis *GetCurrentTask, GetMessage, PeekMessage*

PostMessage

Syntax
```
BOOL PostMessage(HWND Wnd, WORD Msg, WORD wParam,
LONG lParam);
```

Beschreibung Diese Funktion plaziert eine Botschaft in die Anwendungswarteschlange eines Fensters und kehrt dann zurück, ohne darauf zu warten, daß das Fenster die Botschaft bearbeitet. Die übermittelte Botschaft kann durch die Funktionen *GetMessage* oder *PeekMessage* abgerufen werden.

Wnd: Bezeichnet das Fenster, das die Botschaft empfangen soll. Wenn der Parameter *Wnd* 0xFFFF ist, wird die Botschaft an alle überlappten oder Pop-up-Fenster im System gesendet. Die Botschaft wird nicht an untergeordnete Fenster übergeben.

Msg: Bestimmt den Typ der übermittelten Botschaft.

wParam: Bestimmt zusätzliche 16-Bit Botschaftsinformationen.

lParam: Bestimmt zusätzliche 32-Bit Botschaftsinformationen.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Botschaft übermittelt wurde, andernfalls ist er Null.

Hinweis Eine Anwendung sollte die Funktion *PostMessage* nie verwenden, um eine Botschaft an ein Steuerelement zu senden. Wenn ein System, in dem Windows läuft, für Expanded Memory (EMS) konfiguriert ist und eine Anwendung mit der Funktion *PostMessage* eine Botschaft mit verknüpften Daten (auf die der Parameter *lParam* zeigt) an eine zweite Anwendung sendet, muß die erste Anwendung diese Daten in dem globalen Speicherbereich plazieren, der von der Funktion *GlobalAlloc* mit dem Flag *GMEM_LOWER* reserviert wurde. Diese Speicherreservierung ist nur notwendig, wenn *lParam* einen Zeiger enthält.

Im Gegensatz zu anderen Windows-Funktionen kann eine Anwendung *PostMessage* auf Hardware-Interrupt-Ebene aufrufen.

PostQuitMessage

Syntax
```
void PostQuitMessage(int ExitCode);
```

Beschreibung Diese Funktion informiert Windows darüber, daß die Anwendung ihre Ausführung beenden möchte. Sie wird üblicherweise als Antwort auf eine Botschaft *WM_DESTROY* verwendet.

PostQuitMessage übermittelt der Anwendung eine Botschaft *WM_QUIT*, die dem System mitteilt, daß die Anwendung demnächst beenden möchte, und kehrt sofort zurück.

Wenn die Anwendung die Botschaft *WM_QUIT* empfängt, sollte sie die Botschaftsschleife in der Hauptfunktion verlassen und die Steuerung an Windows zurückgeben. Der Exit-Code, der an Windows zurückgegeben wird, muß der Parameter *wParam* der Botschaft *WM_QUIT* sein.

ExitCode: Bestimmt einen Exit-Code der Anwendung. Er wird als Parameter *wParam* in der Botschaft *WM_QUIT* verwendet.

PtInRect

Syntax
```
BOOL PtInRect(LPRECT Rect, POINT Point);
```

Beschreibung Diese Funktion bestimmt, ob der angegebene Punkt innerhalb eines vorgegebenen Rechtecks liegt. Ein Punkt liegt innerhalb eines Rechtecks, wenn er auf dem linken oder oberen Rand oder innerhalb aller vier Kanten liegt. Ein Punkt auf dem rechten oder unteren Rand liegt außerhalb des Rechtecks.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die das Rechteck enthält.

Point: Eine *Point*-Datenstruktur, die den Punkt enthält.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der Punkt innerhalb des angegebenen Rechtecks liegt, andernfalls ist er Null.

PtInRegion

Syntax
```
BOOL PtInRegion(HRGN Rgn, int X, int Y);
```

Beschreibung Diese Funktion ermittelt, ob der durch die Parameter *X* und *Y* bestimmte Punkt in der angegebenen Region liegt.

Rgn: Bezeichnet die Region, die überprüft werden soll.

X: Bestimmt die virtuelle x-Koordinate des Punktes.

Y: Bestimmt die virtuelle y-Koordinate des Punktes.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der Punkt in der Region liegt, andernfalls ist er Null.

PtVisible

Syntax
```
BOOL PtVisible(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion ermittelt, ob der angebene Punkt innerhalb der Clipping-Region des vorgegebenen Gerätekontexts liegt.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate des Punktes.

Y: Bestimmt die virtuelle y-Koordinate des Punktes.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der Punkt innerhalb der Clipping-Region liegt, andernfalls ist er Null.

ReadComm

Syntax
```
int ReadComm(int Cid, LPSTR Buf, int Size);
```

Beschreibung Diese Funktion liest die durch den Parameter *Size* bestimmte Anzahl von Zeichen aus der durch den Parameter *Cid* bezeichneten Schnittstelle und kopiert die Zeichen in den Puffer, auf den der Parameter *Buf* zeigt.

Cid: Bestimmt die Schnittstelle, von der gelesen werden soll. Die Funktion *OpenComm* gibt diesen Wert zurück.

Buf: Zeigt auf den Puffer, der die gelesenen Zeichen empfangen soll.

Size: Bestimmt die Zahl der zu lesenden Zeichen.

Rückgabewert Der Rückgabewert gibt die Zahl der tatsächlich gelesenen Zeichen an. Er ist kleiner als die durch *Size* angegebene Zahl, wenn die Anzahl der Zeichen in der Empfangswarteschlange kleiner als diese ist. Ist er gleich *Size*, können zusätzliche Zeichen für das Gerät in die Warteschlange gestellt worden sein. Sind keine Zeichen vorhanden, ist der Rückgabewert 0.

Tritt ein Fehler auf, wird der Rückgabewert auf einen Wert kleiner 0 gesetzt, wobei der Absolutbetrag die Zahl der bis zum Auftreten des Fehlers gelesenen Zeichen darstellt. Die Fehlerursache kann mit der Funktion *GetCommError* bestimmt werden, welche Fehlercode und -status liefert. Da Fehler auftreten können, wenn keine Bytes vorhanden sind, der Rückgabewert also 0 ist, sollte die Funktion *GetCommError* auch in diesem Fall benutzt werden, um sicherzustellen, daß kein Fehler aufgetreten ist.

Für parallele I/O-Anschlüsse ist der Rückgabewert immer 0.

Querverweis *GetCommError, OpenComm*

RealizePalette

Syntax
```
WORD RealizePalette(HDC DC);
```

Beschreibung Diese Funktion bildet die Systempaletteneinträge in der aktuell ausgewählten virtuellen Palette in einen Gerätekontext ab.

Eine virtuelle Farbpalette fungiert als Puffer zwischen farbintensiven Anwendungen und dem System. Sie erlaubt einer Anwendung, so viele Farben wie nötig zu verwenden, ohne daß dadurch die eigene Farbdarstellung oder Farben, die von anderen Fenstern dargestellt werden, überlagert werden.

Wenn ein Fenster den Fokus hat und *RealizePalette* aufruft, stellt Windows sicher, daß alle angeforderten Farben bis zur Maximalzahl gleichzeitig verfügbarer Farben auf dem Bildschirm dargestellt werden. Zusätzliche Farben zeigt es an, indem diese an verfügbare Farben angeglichen werden. Außerdem gleicht Windows die Farben inaktiver Fenster, die *RealizePalette* aufrufen, so weit wie möglich an die verfügbaren Farben an, womit nicht erwünschte Veränderungen der Farben, die in inaktiven Fenstern dargestellt werden, deutlich verringert werden.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert bestimmt, wieviele Einträge in der virtuellen Palette auf Einträge in der Systempalette abgebildet wurden. Das entspricht der Zahl der Einträge, die diese Funktion neu abgebildet hat, um sich den Änderungen in der Systempalette seit der letzten Realisierung der virtuellen Palette anzupassen.

Rectangle

Syntax
```
BOOL Rectangle(HDC DC, int X1, int Y1, int X2, int Y2);
```

Beschreibung Diese Funktion zeichnet ein Rechteck. Das Innere des Rechtecks wird mit dem gewählten Pinsel ausgefüllt und die Umrandung mit dem gewählten Stift gezeichnet.

DC: Bezeichnet den Gerätekontext.

X1: Bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Rechtecks.

Y1: Bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Rechtecks.

X2: Bestimmt die virtuelle x-Koordinate der unteren rechten Ecke des Rechtecks.

Y2: Bestimmt die virtuelle y-Koordinate der unteren rechten Ecke des Rechtecks.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Rechteck gezeichnet wurde, andernfalls ist er Null.

Hinweis Die Breite des Rechtecks *X2 - X1* darf 32767 Einheiten nicht überschreiten. Dieser Grenzwert gilt auch für die Höhe des Rechtecks (*Y2 - Y1*).

Die aktuelle Position wird von dieser Funktion weder verwendet noch aktualisiert.

Querverweis *FrameRect*

RectInRegion

Syntax
```
BOOL RectInRegion(HRGN Rgn, LPRECT Rect);
```

Beschreibung Diese Funktion bestimmt, ob ein Teil des durch den Parameter *Rect* bezeichneten Rechtecks innerhalb der Grenzen einer Region liegt, die durch den Parameter *Rgn* bezeichnet ist.

Rgn: Bezeichnet die Region.

Rect: Zeigt auf eine *RECT*-Datenstruktur die die Rechteckkoordinaten enthält.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn ein Teil des angegebenen Rechtecks innerhalb der Grenzen der Region liegt, andernfalls ist er Null.

RectVisible

Syntax
```
BOOL RectVisible(HDC DC, LPRECT Rect);
```

Beschreibung Diese Funktion stellt fest, ob ein Teil des gegebenen Rechtecks innerhalb der Clipping-Region des angegebenen Bildschirmkontexts liegt.

DC: Bezeichnet den Gerätekontext.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die virtuellen Koordinaten des Rechtecks enthält.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn ein Teil des Rechtecks *Rect* innerhalb der Clipping-Region liegt, andernfalls ist er Null.

RegisterClass

Syntax
```
BOOL RegisterClass(LPWNDCLASS WndClass);
```

Beschreibung Diese Funktion registriert eine Fensterklasse zur nachfolgenden Verwendung in Aufrufen der Funktion *CreateWindow*. Die Fensterklasse hat die Attribute, die durch die Datenstruktur definiert werden, auf die der Parameter *WndClass* zeigt.

Sollen zwei Klassen mit demselben Namen registriert werden, mißlingt der zweite Versuch und die Informationen für diese Klasse werden ignoriert.

WndClass: Zeigt auf eine *WNDCLASS*-Datenstruktur. Die Struktur muß mit den geeigneten Klassenattributen gefüllt sein, bevor sie an die Funktion übergeben wird. Im folgenden Abschnitt werden die Details erläutert.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Klasse registriert wurde, andernfalls ist er Null.

Callback-Funktion
```
BOOL FAR PASCAL WndProc(HWND Wnd, WORD Msg,
WORD wParam, DWORD lParam);
```

WndProc: Ein Platzhalter für den anwendungsunterstützten Funktionsnamen. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Wnd: Bezeichnet das Fenster, das die Botschaft empfängt.

Msg: Bestimmt die Botschaftsnummer.

wParam: Bestimmt zusätzliche botschaftsabhängige Information.

lParam: Bestimmt zusätzliche botschaftsabhängige Information.

Rückgabewert Die Fensterfunktion gibt das Ergebnis der Botschaftsverarbeitung zurück. Die möglichen Rückgabewerte hängen von der gesendeten Botschaft ab.

RegisterClipboardFormat

Syntax
```
WORD RegisterClipboardFormat(LPSTR FormatName);
```

Beschreibung Diese Funktion registriert ein neues Zwischenablageformat, auf dessen Namen der Parameter *FormatName* zeigt. Das registrierte Format kann in darauffolgenden Zwischenablagefunktionen als gültiges Format, in dem Daten übergeben werden können, verwendet werden. Es erscheint in der Formatliste der Zwischenablage.

FormatName: Zeigt auf einen nullterminierten String, der das neue Format benennt.

Rückgabewert Der Rückgabewert beschreibt das neu registrierte Format. Wenn der identische Formatname schon vorher registriert wurde (z.B. durch eine andere Anwendung), wird der Referenzzähler des Formats erhöht und derselbe Wert zurückgegeben, als wäre das Format erstmals registriert worden. Der Rückgabewert ist 0, wenn das Format nicht registriert werden kann.

Hinweis Der Formatwert, den die Funktion *RegisterClipboardFormat* zurückgibt, liegt im Bereich von 0xC000 bis 0xFFFF.

RegisterWindowMessage

Syntax
```
WORD RegisterWindowMessage(LPSTR Str);
```

Beschreibung Diese Funktion erzeugt eine systemweit eindeutige Fensterbotschaft. Der zurückgegebene Botschaftswert kann für Aufrufe der Funktionen *SendMessage* oder *PostMessage* verwendet werden. *RegisterWindowMessage* wird üblicherweise für den Datenaustausch zwischen zwei zusammenarbeitenden Anwendungen genutzt.

Wird derselbe Botschaftsstring von zwei unterschiedlichen Anwendungen registriert, so wird auch derselbe Botschaftswert zurückgegeben. Die Botschaft bleibt registriert, bis der Anwender die Windows-Sitzung beendet.

Str: Zeigt auf den Botschaftsstring, der registriert werden soll.

Rückgabewert Bei erfolgreicher Ausführung ist der Rückgabewert ein WORD im Bereich zwischen 0xC000 bis 0xFFFF, andernfalls ist er 0.

Hinweis *RegisterWindowMessage* sollte nur benutzt werden, wenn dieselbe Nachricht von mehr als einer Anwendung verstanden werden muß. Um private Botschaften innerhalb einer Anwendung zu senden, kann die Anwendung ein WORD im Bereich von *WM_USER* bis 0xBFFF verwenden.

ReleaseCapture

Syntax
```
void ReleaseCapture(void);
```

Beschreibung Hebt die exklusive Benutzung der Maus (Mausfang) auf und stellt die normale Eingabeverarbeitung wieder her.

Die Funktion hat keine Parameter.

Rückgabewert Keiner.

Hinweis Eine Anwendung ruft diese Funktion nach einem Aufruf der Funktion *SetCapture* auf.

ReleaseDC

Syntax `int ReleaseDC(HWND Wnd, HDC DC);`

Beschreibung Diese Funktion hebt einen Gerätekontext auf und gibt ihn zur Verwendung durch andere Anwendungen frei.

Die Wirkung von *ReleaseDC* hängt vom Typ des Gerätekontexts ab. Sie gibt nur allgemeine und Fenster-Gerätekontexte frei und hat keinen Einfluß auf Klassen- oder private Gerätekontexte.

Wnd: Bezeichnet das Fenster, dessen Gerätekontext aufgehoben werden soll.

DC: Bezeichnet den aufzuhebenden Gerätekontext.

Rückgabewert Der Rückgabewert ist 1, wenn der Gerätekontext aufgehoben wurde, andernfalls ist er 0.

Hinweis Die Anwendung muß *ReleaseDC* für jeden Aufruf von *GetWindowDC* und für solche Aufrufe von *GetDC*, die einen gemeinsamen Gerätekontext liefern, aufrufen.

Querverweis *GetDC, GetWindowDC*

RemoveFontResource

Syntax `BOOL RemoveFontResource(LPSTR FileName);`

Beschreibung Diese Funktion löscht eine hinzugefügte Schriftressource aus der Datei, die durch den Parameter *FileName* benannt ist, oder aus der Windows-Schrifttabelle.

FileName: Zeigt auf einen String, der die Schriftressourcendatei benennt oder auf ein Handle zu einem geladenen Modul. Zeigt *FileName* auf den Dateinamen der Schriftressource, muß der String nullterminiert sein und dem DOS-Dateinamensformat entsprechen.

Wenn *FileName* ein Handle enthält, muß dieses im niederwertigen WORD liegen und das höherwertige WORD muß 0 sein.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Jede Anwendung, die Schriften aus der Windows-Schriftentabelle löscht oder ihr hinzufügt, sollte andere Fenster mit der Funktion *SendMessage* davon benachrichtigen. Dabei sollte der Parameter *Wnd* auf -1 gesetzt sein, um eine Botschaft *WM_FONTCHANGE* an alle Hauptfenster im System zu senden. *RemoveFontResource* löscht die Schriftressource nicht unmittelbar. Wenn es noch ausstehende Referenzen zu ihr

gibt, bleibt sie geladen, bis mit der Funktion *DeleteObject* die letzte sich auf sie beziehende virtuelle Schrift gelöscht wurde.

Querverweis *DeleteObject, WM_FONTCHANGE*

RemoveMenu

Syntax
```
BOOL RemoveMenu(HMENU Menu, WORD Position, WORD Flags);
```

Beschreibung Diese Funktion löscht einen Menüeintrag mit verbundenem Pop-up-Menü aus dem durch den Parameter *Menu* bezeichneten Menü. Dabei gibt sie das Handle auf das Pop-up-Menü nicht frei und erlaubt somit die Wiederverwendung des Menüs.

Vor dem Aufruf dieser Funktion sollte die Anwendung *GetSubMenu* aufrufen, um das Handle des Pop-up-Menüs abzurufen.

Menu: Bezeichnet das Menü, das verändert werden soll.

Position: Bestimmt die zu löschende Menüoption. Die Interpretation des Parameters *Position* hängt vom Wert des Parameters *Flags* ab.

Flags	Position
MF_BYCOMMAND	Bestimmt die Befehls-ID der existierenden Menüoption.
MF_BYPOSITION	Bestimmt die Position der Menüoption. Die erste Option im Menü steht an Position 0.

Flags: Bestimmt, wie der Parameter *Position* interpretiert wird.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Nach jeder Änderung eines Menüs (unabhängig davon, ob es sich im angezeigten Fenster befindet oder nicht), sollte die Funktion *DrawMenuBar* aufgerufen werden.

Querverweis *DrawMenuBar, GetSubMenu*

RemoveProp

Syntax
```
HANDLE RemoveProp(HWND Wnd, LPSTR Str);
```

Beschreibung Diese Funktion löscht einen Eintrag aus der Eigenschaftsliste des angegebenen Fensters. Der String, der durch den Parameter *Str* bezeichnet ist, bestimmt den zu löschenden Eintrag.

RemoveProp gibt das Daten-Handle zurück, das mit dem String verbunden ist, so daß die Anwendung die Daten, die mit dem Handle verbunden sind, freigeben kann.

Wnd: Bezeichnet das Fenster, dessen Eigenschaftsliste geändert werden soll.

Str: Zeigt auf einen nullterminierten String oder auf ein Atom, das einen String bezeichnet. Wenn ein Atom angegeben ist, dann muß es vorher mittels der Funktion *AddAtom* erzeugt worden sein. Das Atom, ein 16-Bit-Wert, muß im niederwertigen WORD von *Str* plaziert werden; das höherwertige WORD muß 0 sein.

Rückgabewert Der Rückgabewert ist 0, wenn der String in der angegebenen Eigenschaftsliste nicht gefunden werden kann. Andernfalls wird das Daten-Handle des Strings zurückgegeben.

Hinweis Eine Anwendung muß die Daten-Handle, die gelöschten Einträgen aus einer Eigenschaftsliste zugeordnet sind, freigeben. Die Anwendung sollte nur diejenigen Eigenschaften löschen, die sie in die Eigenschaftsliste eingefügt hatte.

ReplyMessage

Syntax `void ReplyMessage(LONG Reply);`

Beschreibung Diese Funktion wird verwendet, um eine mit der Funktion *SendMessage* übergebene Botschaft zu beantworten, ohne die Steuerung an die von *SendMessage* aufgerufene Funktion zurückzugeben.

Der Aufruf dieser Funktion erlaubt es den Aufrufern von *SendMessage* und *ReplyMessage*, mit der Bearbeitung fortzufahren.

Normalerweise fährt eine Task, die *SendMessage* aufruft, um eine Botschaft an eine andere Task zu senden, solange nicht mit der Ausführung fort, bis die Fensterfunktion antwortet, die Windows aufruft, um die Botschaft zu empfangen. Muß jedoch eine Task, die zum Empfangen einer Botschaft aufgerufen wurde, bestimmte Operationen ausführen, die die Steuerung abgeben könnten (etwa die Funktionen *MessageBox* oder *DialogBox*), könnte Windows blockiert werden. Der Grund liegt darin, daß die sendende Task Botschaften zur Ausführung und Verarbeitung benötigt, diese aber nicht erhält, weil sie auf die Rückkehr von *SendMessage* wartet.

Eine Anwendung kann dieses Problem umgehen, indem die Task, die die Botschaft empfängt, *ReplyMessage* aufruft, bevor sie eine Operation ausführt, die eine Abgabe der Steuerung bewirken könnte.

ReplyMessage hat keine Wirkung, wenn die Botschaft nicht durch *SendMessage* oder wenn sie durch dieselbe Task gesendet wurde.

Reply: Das Ergebnis der Verarbeitung der Botschaft. Die möglichen Werte hängen von der tatsächlich gesendeten Botschaft ab.

Rückgabewert Keiner.

ResizePalette

Syntax
```
BOOL ResizePalette(HPALETTE Palette, WORD NumEntries);
```

Beschreibung Diese Funktion ändert die Größe der virtuellen Palette, die durch den Parameter *Palette* bestimmt ist, in die Anzahl von Einträgen, die durch den Parameter *NumEntries* vorgegeben wird.

Ruft eine Anwendung *ResizePalette* auf, um die Palette zu verkleinern, werden die Einträge, die in der Palette mit der neuen Größe zurückbleiben, nicht verändert. Ruft die Anwendung *ResizePalette* auf, um sie zu vergrößern, werden die zusätzlichen Paletteneinträge auf Schwarz gesetzt (die Rot-, Grün- und Blauwerte sind alle 0). Außerdem werden die Flags für alle zusätzlichen Einträge auf 0 gesetzt.

Palette: Bezeichnet die Palette, die verändert werden soll.

NumEntries: Bestimmt die Zahl der Einträge in der Palette, nachdem ihre Größe neu festgelegt wurde.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Größe der Palette erfolgreich geändert wurde, andernfalls ist er Null.

RestoreDC

Syntax
```
BOOL RestoreDC(HDC DC, int SaveDC);
```

Beschreibung Diese Funktion stellt den mit *DC* angegebenen Gerätekontext entsprechend dem durch *SaveDC* bezeichneten vorherigen Status wieder her. *RestoreDC* stellt durch Kopieren von Statusinformationen, die durch vorausgegangene Aufrufe der Funktion *SaveDC* auf dem Kontext-Stack abgelegt wurden, den Gerätekontext wieder her.

Der Kontext-Stack kann Statusinformationen für mehrere Gerätekontexte enthalten. Wenn der durch *SaveDC* angegebene Kontext nicht oben auf dem Stack abgelegt ist, löscht *RestoreDC* jede Statusinformation zwischen dem durch *SaveDC* bezeichneten Gerätekontext und dem oberen Ende des Stacks, diese Informationen sind somit verloren.

DC: Bezeichnet den Gerätekontext.

SaveDC: Bestimmt den Gerätekontext, der wiederhergestellt werden soll. Er kann ein Wert sein, der von einem vorange-

gangenen Aufruf der Funktion *SaveC* zurückgegeben wurde. Wenn *SaveDC* -1 ist, wird der zuletzt gesicherte Gerätekontext wiederhergestellt.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der angegebene Gerätekontext wiederhergestellt wurde, ist dies nicht der Fall, ist er Null.

RGB

Syntax
```
DWORD RGB(BYTE Red, BYTE Green, BYTE Blue);
```

Beschreibung Dieses Makro wählt aufgrund der gegebenen Parameter und der Farbwiedergabemöglichkeiten des Ausgabegeräts eine RGB-Farbe aus.

Red: Bestimmt die Intensität des roten Farbfeldes.

Green: Bestimmt die Intensität des grünen Farbfeldes.

Blue: Bestimmt die Intensität des blauen Farbfeldes.

Rückgabewert Der Rückgabewert ist die resultierende RGB-Farbe.

Hinweis Die Intensität für jedes Argument kann von 0 bis 255 reichen. Wenn alle drei Intensitätswerte mit 0 angegeben sind, ist das Ergebnis Schwarz. Wenn alle drei Intensitätswerte mit 255 angegeben werden, ist das Ergebnis Weiß.

Querverweis *PaletteIndex*, *PaletteRGB*.

RoundRect

Syntax
```
BOOL RoundRect(HDC DC, int X1, int Y1, int X2, int Y2,
int X3, int Y3);
```

Beschreibung Diese Funktion zeichnet Rechtecke mit abgerundeten Ecken. Das Innere des Rechtecks wird mit dem gewählten Pinsel ausgefüllt, die Umrandung wird mit dem ausgewählten Stift gezeichnet.

DC: Bezeichnet den Gerätekontext.

X1, Y1: Bestimmt die virtuelle x- bzw. y-Koordinate der oberen linken Ecke des Rechtecks.

X2, Y2: Bestimmt die virtuelle x- bzw. y-Koordinate der unteren rechten Ecke des Rechtecks.

X3, Y3: Bestimmt die Breite bzw. Höhe der Ellipse, die verwendet wird, um die abgerundeten Ecken zu zeichnen.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Rechteck gezeichnet wurde, andernfalls ist er Null.

Hinweis Die Breite des Rechtecks, die sich aus der Differenz *X2-X1* ergibt, darf 32767 Einheiten nicht überschreiten. Dieser Grenzwert gilt auch für die Höhe des Rechtecks (*Y2-Y1*).

Die aktuelle Position wird von dieser Funktion weder benutzt noch aktualisiert.

SaveDC

Syntax
```
int SaveDC(HDC DC);
```

Beschreibung Diese Funktion speichert den aktuellen Status des durch den Parameter *DC* bezeichneten Gerätekontexts, indem sie Statusinformationen (zum Beispiel die Clipping-Region, ausgewählte Objekte und den Abbildungsmodus) auf einen Kontext-Stack kopiert. Der gespeicherte Gerätekontext kann anschliessend mit *RestoreDC* wiederhergestellt werden.

DC: Bezeichnet den Gerätekontext, der gespeichert werden soll.

Rückgabewert Der Rückgabewert bezeichnet nach erfolgreicher Ausführung den gespeicherten Gerätekontext, andernfalls ist er 0.

Hinweis Die Funktion *SaveDC* kann beliebig oft verwendet werden, um den Status beliebig vieler Gerätekontexte zu speichern.

Querverweis *RestoreDC*

ScaleViewportExt

Syntax
```
DWORD ScaleViewportExt(HDC DC, int Xnum, int Xdenom,
int Ynum, int Ydenom);
```

Beschreibung Diese Funktion verändert die Abmessungen des Grafikfensters (VE) relativ zu den aktuellen Werten.

Dabei gilt folgende Formel:

xNeuVE = (xAltVE * *Xnum*) / *Xdenom*
yNeuVE = (yAltVE * *Ynum*) / *Ydenom*

Die neuen Abmessungen werden berechnet, indem die aktuellen Abmessungen erst mit dem angegebenen Zähler multipliziert und dann durch den angegebenen Nenner dividiert werden.

DC: Bezeichnet den Gerätekontext.

Xnum: Bestimmt den Wert, mit dem die aktuelle x-Abmessung multipliziert werden soll (Zähler).

Xdenom: Bestimmt den Wert, durch den die aktuelle x-Abmessung dividiert werden soll (Nenner).

Ynum: Bestimmt den Wert, mit dem die aktuelle y-Abmessung multipliziert werden soll.

Ydenom: Bestimmt den Wert, durch den die aktuelle y-Abmessung dividiert werden soll.

Rückgabewert Der Rückgabewert gibt die vorherigen Grafikfensterabmessungen in Geräteeinheiten an. Die vorherige y-Abmessung liegt im höherwertigen, die vorherige x-Abmessung im niederwertigen WORD.

ScaleWindowExt

Syntax
```
DWORD ScaleWindowExt(HDC DC, int Xnum, int Xdenom,
int Ynum, int Ydenom);
```

Beschreibung Diese Funktion verändert die Abmessungen des Fensters (WE) relativ zu den aktuellen Werten.

Dabei gilt folgende Formel:

xNeu WE = (xAlt WE * *Xnum*) / *Xdenom*
yNeu WE = (yAlt WE * *Ynum*) / *Ydenom*

Die neuen Abmessungen werden berechnet, indem die aktuellen Abmessungen erst mit dem angegebenen Zähler multipliziert und dann durch den angegebenen Nenner dividiert werden.

DC: Bezeichnet den Gerätekontext

Xnum: Bestimmt den Wert, mit dem die aktuelle x-Abmessung multipliziert werden soll (Zähler).

Xdenom: Bestimmt den Wert, durch den die aktuelle x-Abmessung dividiert werden soll (Nenner).

Ynum: Bestimmt den Wert, mit dem die aktuelle y-Abmessung multipliziert werden soll.

Ydenom: Bestimmt den Wert, durch den die aktuelle y-Abmessung dividiert werden soll.

Rückgabewert Der Rückgabewert gibt die vorherigen Fensterabmessungen in virtuellen Einheiten an. Die vorherige y-Abmessung liegt im höherwertigen, die vorherige x-Abmessung im niederwertigen WORD.

ScreenToClient

Syntax
```
void ScreenToClient(HWND Wnd, LPPOINT Point);
```

Beschreibung Diese Funktion konvertiert die Bildschirmkoordinaten eines gegebenen Punktes auf dem Bildschirm in Client-Koordinaten.

ScreenToClient verwendet das Fenster, das durch den Parameter *Wnd* bezeichnet ist und die Bildschirmkoordinaten, die in der durch *Point* bezeichneten *POINT*-Datenstruktur angegeben sind, um die Client-Koordinaten zu berechnen. Sie ersetzt dann die Bildschirmkoordinaten durch die Client-Koordinaten. Die neuen Koordinaten gelten relativ zur linken oberen Ecke des Client-Bereichs des angegebenen Fensters.

Wnd: Bezeichnet das Fenster, dessen Client-Bereich für die Konvertierung verwendet wird.

Point: Zeigt auf eine *POINT*-Datenstruktur, die die zu konvertierenden Bildschirmkoordinaten enthält.

Rückgabewert Keiner.

Hinweis Die Formel in *ScreenToClient* setzt voraus, daß der Punkt in Bildschirmkoordinaten angegeben ist.

ScrollDC

Syntax
```
BOOL ScrollDC(HDC DC, int dx, int dy, LPRECT Scroll,
LPRECT Clip, HRGN UpdateRgn, LPRECT UpdateRect);
```

Beschreibung Diese Funktion rollt ein Rechteck aus Bits in horizontaler und vertikaler Richtung.

Der Parameter *Scroll* zeigt auf das Rechteck, das gerollt werden soll. Der Parameter *dx* bezeichnet die Zahl der Einheiten, um die horizontal, *dy* die Zahl der Einheiten, um die vertikal gerollt werden soll.

DC: Bezeichnet den Gerätekontext, der die Bits enthält, die gerollt werden sollen.

dx: Bestimmt die Zahl der horizontalen Bildlaufeinheiten.

dy: Bestimmt die Zahl der vertikalen Bildlaufeinheiten.

Scroll: Zeigt auf eine *RECT*-Datenstruktur mit den Koordinaten des zu rollenden Rechtecks.

Clip: Zeigt auf eine *RECT*-Datenstruktur mit den Koordinaten des Clipping-Rechtecks. Wenn dieses Rechteck kleiner ist als das durch *Scroll* bezeichnete Original, findet der Bildlauf nur in dem kleineren Rechteck statt.

UpdateRgn: Bezeichnet die Region, die vom Bildlauf beeinflußt wird. *ScrollDC* definiert diese Region, die nicht unbedingt ein Rechteck ist.

UpdateRect: Zeigt auf die *RECT*-Datenstruktur, die bei der Rückkehr die Koordinaten des Rechtecks enthält, das den Bildlaufbereich umfaßt. Dies ist der größte rechteckige Bereich, der ein Neuzeichnen erfordert.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der Bildlauf ausgeführt wurde, andernfalls ist er Null.

Hinweis Windows berechnet das Aktualisierungsrechteck nicht, wenn der Parameter *UpdateRect* NULL ist. Die Aktualisierungsregion wird nicht berechnet, wenn sowohl *UpdateRect* als auch *UpdateRgn* NULL sind. Ist *UpdateRgn* nicht NULL, nimmt Windows an, daß er ein gültiges Region-Handle zu der Region enthält, die vom Bildlauf betroffen wird (definiert durch die Funktion *ScrollDC*).

Wenn es notwendig ist, den gesamten Client-Bereich eines Fensters zu rollen, sollte eine Anwendung die Funktion *ScrollWindow* verwenden.

ScrollWindow

Syntax
```
void ScrollWindow(HWND Wnd, int XAmount, int YAmount,
LPRECT Rect, LPRECT ClipRect);
```

Beschreibung Diese Funktion verschiebt ein Fenster, indem sie die Inhalte des Client-Bereichs des Fensters um die durch den Parameter *XAmount* vorgegebene Anzahl von Einheiten entlang der x-Achse und um die durch den Parameter *YAmount* vorgegebene Anzahl von Einheiten entlang der y-Achse des Bildschirms bewegt.

Der Bildlauf erfolgt nach rechts, wenn *XAmount* positiv ist und nach links, wenn dieser Parameter negativ ist. Der Bildlauf erfolgt nach unten, wenn der Parameter *YAmount* positiv ist und nach oben, wenn er negativ ist.

Wnd: Bezeichnet das Fenster, für dessen Client-Bereich ein Bildlauf durchgeführt werden soll.

XAmount: Bestimmt den Betrag in gerätebezogenen Einheiten, um den in x-Richtung gerollt werden soll.

YAmount: Bestimmt den Betrag in gerätebezogenen Einheiten, um den in y-Richtung gerollt werden soll.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die den Teil des Client-Bereiches bezeichnet, der gerollt werden soll. Ist *Rect* NULL, wird der gesamte Client-Bereich gerollt.

ClipRect: Zeigt auf eine *RECT*-Datenstruktur, die das Clipping-Rechteck bezeichnet, welches gerollt werden soll. Nur Bits innerhalb des Rechtecks werden gerollt. Ist *ClipRect* NULL, findet der Bildlauf im gesamten Fenster statt.

Rückgabewert Keiner.

Hinweis Befindet sich das Caret in dem Fenster, das gerade gerollt wird, blendet *ScrollWindow* das Caret automatisch aus, um zu ver-

hindern, daß es gelöscht wird. Nach Beendigung des Bildlaufs wird das Caret wieder hergestellt, die Position des Caret wird entsprechend angeglichen.

Der von *ScrollWindow* nicht abgedeckte Bereich wird nicht neu gezeichnet, sondern mit der Aktualisierungsregion des Fensters kombiniert. Die Anwendung empfängt eventuell eine Botschaft *WM_PAINT*, die sie davon in Kenntnis setzt, daß die Region neu gezeichnet werden muß. Um den nicht abgedeckten Bereich zur selben Zeit neu zu zeichnen, zu der der Bildlauf durchgeführt wird, sollte *UpdateWindow* sofort nach dem Aufruf von *ScrollWindow* aufgerufen werden. Wenn der Parameter *Rect* NULL ist, werden die Positionen aller untergeordneter Fenster im Fenster um den Betrag verschoben, den *XAmount* und *YAmount* vorgeben; sämtliche unvollständigen (nicht gezeichneten) Bereiche im Fenster werden ebenso verschoben. *ScrollWindow* arbeitet schneller, wenn *Rect* NULL ist.

Ist der Parameter *Rect* nicht NULL, werden die Positionen untergeordneter Fenster nicht verändert und ungültige Bereiche im Fenster werden nicht verschoben. Um hier Probleme bei der Aktualisierung zu vermeiden, sollte man *UpdateWindow* aufrufen, um das Fenster vor dem Aufruf von *ScrollWindow* neu zu zeichnen.

Querverweis *UpdateWindow, WM_PAINT*

SelectClipRgn

Syntax
```
int SelectClipRgn(HDC DC, HRGN Rgn);
```

Beschreibung Diese Funktion wählt die angegebene Region als die aktuelle Clipping-Region für den bezeichneten Gerätekontext. Es wird nur eine Kopie der ausgewählten Region benutzt. Die Region selbst kann für eine beliebige Anzahl von anderen Gerätekontexten ausgewählt oder gelöscht werden.

DC: Bezeichnet den Gerätekontext

Rgn: Bezeichnet die Region, die ausgewählt werden soll.

Rückgabewert Der Rückgabewert bestimmt den Typ der Region. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
COMPLEXREGION	Neue Clipping-Region hat überlappende Grenzen.
ERROR	Gerätekontext oder Region-Handle nicht gültig.
NULLREGION	Neue Clipping-Region ist leer.
SIMPLEREGION	Neue Clipping-Region hat keine überlappenden Grenzen.

Hinweis

SelectClipRgn geht davon aus, daß die Koordinaten der vorgegebenen Region in Geräteeinheiten angegeben sind.

Einige Druckgeräte unterstützen die Grafikdarstellung mit niedrigeren Auflösungen als Textdarstellung, um die Geschwindigkeit bei gleichzeitiger Qualitätseinbuße zu erhöhen. Diese Geräte skalieren Koordinaten für Grafiken so, daß ein Grafikpunkt auf dem Druckgerät zwei oder vier echten Gerätepunkten entspricht. Dieser Skalierungsfaktor beeinflußt das Clipping. Wenn eine Region verwendet werden soll, um Grafiken auszuschneiden, müssen seine Koordinaten durch den Skalierungsfaktor geteilt werden. Soll die Region verwendet werden, um Text auszuschneiden, ist keine Angleichung der Skalierung notwendig. Der Skalierungsfaktor wird durch die Druckersteuersequenz *GETSCALINGFACTOR* bestimmt.

SelectObject

Syntax

```
HANDLE SelectObject(HDC DC, HANDLE Object);
```

Beschreibung

Diese Funktion wählt das virtuelle Objekt, das der Parameter *Object* bezeichnet, als aktuelles Objekt des angegebenen Gerätekontexts. Das neue Objekt ersetzt das vorherige Objekt gleichen Typs. Ist zum Beispiel *Object* das Handle zu einem virtuellen Stift, ersetzt die Funktion *SelectObject* den ausgewählten Stift durch den von *Object* vorgegebenen Stift.

Ausgewählte Objekte sind die Standardobjekte, die von den GDI-Ausgabefunktionen benutzt werden, um Linien zu zeichnen, Innenflächen auszufüllen, Text zu schreiben und Ausgaben auf spezielle Bereiche der Bildschirmoberfläche zu tätigen. Obwohl ein Gerätekontext sechs ausgewählte Objekte haben kann (Pinsel, Stift, Schrift, Bitmap, Region und virtuelle Palette), kann nicht mehr als ein Objekt des angegebenen Typs zur gleichen Zeit ausgewählt sein. *SelectObject* wählt keine virtuelle Palette aus, dazu muß *SelectPalette* aufgerufen werden.

DC: Bezeichnet den Gerätekontext.

Object: Bezeichnet das Objekt, das ausgewählt werden soll. Es kann eines der folgenden sein und muß vorher mit einer der nachstehenden Funktionen erzeugt worden sein (Bitmaps können nur für Speichergerätekontexte und immer nur für einen Gerätekontext ausgewählt werden):

Objekt	Funktion
Bitmap	*CreateBitmap* *CreateBitmapIndirect* *CreateCompatibleBitmap* *CreateDIBitmap*

Objekt	Funktion
Pinsel	*CreateBrushIndirect*
	CreateHatchBrush
	CreatePatternBrush
	CreateSolidBrush
Schrift	*CreateFont*
	CreateFontIndirect
Stift	*CreatePen*
	CreatePenIndirect
Region	*CombineRgn*
	CreateEllipticRgn
	CreateEllipticRgnIndirect
	CreatePolygonRgn
	CreateRectRgn
	CreateRectRgnIndirect

Rückgabewert Der Rückgabewert bezeichnet das Objekt, das durch das vom Parameter *Object* vorgegebene Objekt ersetzt wurde. Er ist 0, wenn ein Fehler auftritt. Wenn der Parameter *DC* ein Metafile bezeichnet, ist der Rückgabewert bei erfolgreicher Ausführung der Funktion ungleich 0, andernfalls ist er 0.

Wenn eine Region ausgewählt ist, ist der Rückgabewert der gleiche wie der von *SelectClipRgn*.

Hinweis Wird eine Schrift, ein Stift oder ein Pinsel mit *SelectObject* ausgewählt, reserviert die GDI Speicher für dieses Objekt in ihrem Datensegment. Da der Datensegmentspeicher der GDI begrenzt ist, sollte die Funktion *DeleteObject* verwendet werden, um jedes zeichnende Objekt zu löschen, das nicht mehr benötigt wird. Vor dem Löschen des letzten, nicht benötigten zeichnenden Objekts sollte die Anwendung das ursprüngliche (Standard-)Objekt in den Gerätekontext wählen.

Ein Bitmap kann von der Anwendung zu einem Zeitpunkt nur in einen Gerätekontext ausgewählt werden.

Querverweis *DeleteObject, SelectClipRgn, SelectPalette*

SelectPalette

Syntax
```
HPALETTE SelectPalette(HDC DC, HPALETTE Palette,
BOOL ForceBackground);
```

Beschreibung Diese Funktion wählt die durch den Parameter *Palette* bezeichnete virtuelle Palette als ausgewähltes Palettenobjekt des durch den Parameter *DC* bezeichneten Gerätekontextes aus. Die neue Palette wird das von der GDI bei der Steuerung der im Gerätekontext dargestellten Farben verwendete Palettenobjekt und ersetzt die vorherige Palette.

DC: Bezeichnet den Gerätekontext.

Palette: Bezeichnet die virtuelle Palette, die ausgewählt werden soll. *CreatePalette* erstellt eine virtuelle Palette.

ForceBackground: Bestimmt, ob die virtuelle Palette eine Hintergrundpalette sein soll. Ist der Wert ungleich Null, ist die ausgewählte Palette immer eine Hintergrundpalette, ungeachtet dessen, ob das Fenster den Fokus hat. Ist er Null, ist die virtuelle Palette eine Vordergrundpalette, wenn das Fenster den Fokus hat.

Rückgabewert Der Rückgabewert bezeichnet die virtuelle Palette, die von der durch *Palette* vorgegebenen Palette ersetzt wurde. Er ist 0, wenn ein Fehler vorliegt.

Hinweis Eine Anwendung kann eine virtuelle Palette in mehr als einen Gerätekontext auswählen. Änderungen an einer virtuellen Palette beeinflussen jedoch alle Gerätekontexte, für die sie ausgewählt ist.

Wählt eine Anwendung ein Palettenobjekt für mehr als einen Gerätekontext aus, müssen die Gerätekontexte alle zu demselben physikalischen Gerät gehören (wie etwa einem Bildschirm oder einem Drucker).

Querverweis *CreatePalette*

SendDlgItemMessage

Syntax
```
LONG SendDlgItemMessage(HWND Dlg, int IDDlgItem,
WORD Msg, WORD wParam, LONG lParam);
```

Beschreibung Diese Funktion sendet eine Botschaft an ein durch den Parameter *IDDlgItem* bezeichnetes Steuerelement innerhalb des durch den Parameter *Dlg* bezeichneten Dialogfensters. Die Funktion *SendDlgItemMessage* kehrt nicht zurück, bis die Botschaft verarbeitet wurde.

Dlg: Bezeichnet das Dialogfenster, in dem sich das Steuerelement befindet.

IDDlgItem: Bestimmt den Integerbezeichner des Dialogelements, das die Botschaft empfangen soll.

Msg: Bestimmt den Botschaftswert.

wParam: Bestimmt zusätzliche Botschaftsinformationen.

lParam: Bestimmt zusätzliche Botschaftsinformationen.

Rückgabewert Der Rückgabewert ist der Wert, der von der Fensterfunktion des Steuerelements zurückgegeben wurde oder 0, wenn der Bezeichner des Steuerelements nicht gültig ist.

Hinweis Die Verwendung von *SendDlgItemMessage* ist identisch mit dem Abruf des Handle zu dem angegebenen Steuerelement und einem anschließenden Aufruf der Funktion *SendMessage*.

SendMessage

Syntax
```
LONG SendMessage(HWND Wnd, WORD Msg, WORD wParam,
LONG lParam);
```

Beschreibung Diese Funktion sendet eine Botschaft an ein oder mehrere Fenster. *SendMessage* kehrt solange nicht zurück, bis die Botschaft verarbeitet wurde. Ist das Fenster, das die Botschaft empfängt, Bestandteil der gleichen Anwendung, wird die Fensterfunktion sofort als Subroutine aufgerufen. Ist das Fenster Bestandteil einer anderen Task, schaltet Windows zu dieser Task, ruft die geeignete Fensterfunktion auf und übergibt dann die Botschaft. Die Botschaft wird nicht in der Warteschlange der Zielanwendung plaziert.

Wnd: Bezeichnet das Fenster, das die Botschaft empfangen soll. Ist der Parameter *Wnd* 0xFFFF, wird die Botschaft an alle Pop-up-Fenster im System übergeben. Die Botschaft wird nicht an untergeordnete Fenster gesendet.

Msg: Bestimmt die zu sendende Botschaft.

wParam: Bestimmt zusätzliche Botschaftsinformationen.

lParam: Bestimmt zusätzliche Botschaftsinformationen.

Rückgabewert Der Rückgabewert ist der Wert, der von der Fensterfunktion, die die Botschaft empfangen hat, zurückgegeben wird. Er hängt somit von der gesendeten Botschaft ab.

Hinweis Wenn ein System, in dem Windows läuft, für Expanded Memory (EMS) konfiguriert ist und eine Anwendung mit der Funktion *SendMessage* eine Botschaft mit verknüpften Daten (auf die der Parameter *lParam* zeigt) an eine zweite Anwendung sendet, muß die erste Anwendung diese Daten in dem globalen Speicherbereich plazieren, der von der Funktion *GlobalAlloc* mit dem Flag *GMEM_LOWER* reserviert wurde. Diese Speicherreservierung ist nur notwendig, wenn *lParam* einen Zeiger enthält.

SetActiveWindow

Syntax
```
HWND SetActiveWindow(HWND Wnd);
```

Beschreibung Diese Funktion macht ein Hauptfenster zum aktiven Fenster.

Wnd: Bezeichnet das zu aktivierende Hauptfenster.

Rückgabewert Der Rückgabewert bezeichnet das Fenster, das vorher aktiv war. Die Funktion *SetActiveWindow* sollte mit Vorsicht angewandt werden, da sie einer Anwendung erlaubt, das aktive Fenster und den Fokus unumschränkt zu übernehmen. Normalerweise überwacht Windows jede Aktivierung.

SetBitmapBits

Syntax
```
LONG SetBitmapBits(HPITMAP Bitmap, DWORD Count,
LPSTR Bits);
```

Beschreibung Diese Funktion setzt die Bits eines Bitmaps auf die Werte, die der Parameter *Bits* vorgibt.

Bitmap: Bezeichnet das Bitmap, das gesetzt werden soll.

Count: Bestimmt die Zahl der Bytes, auf die *Bits* zeigt.

Bits: Zeigt auf die Bitmap-Bits, die in einem Byte-Array gespeichert sind.

Rückgabewert Der Rückgabewert gibt bei erfolgreicher Ausführung die Zahl der Bytes an, die beim Setzen der Bitmap-Bits benutzt wurden, andernfalls ist er *Null*.

SetBitmapDimension

Syntax
```
DWORD SetBitmapDimension(HBITMAP ABitmap, int X, int Y);
```

Beschreibung Diese Funktion weist einem Bitmap Breite und Höhe in Einheiten von 0.1 Millimeter zu. Diese Werte werden von der GDI intern nicht genutzt; die Funktion *GetBitmapDimension* kann verwendet werden, um sie abzurufen.

ABitmap: Bezeichnet das Bitmap.

X: Bestimmt die Breite des Bitmaps in Einheiten von 0.1 Millimeter.

Y: Bestimmt die Höhe des Bitmaps in Einheiten von 0.1 Millimeter.

Rückgabewert Der Rückgabewert gibt die vorherigen Abmessungen des Bitmaps an. Die Höhe liegt im höherwertigen, die Breite im niederwertigen WORD.

SetBkColor

Syntax
```
DWORD SetBkColor(HDC DC, COLORREF Color);
```

Beschreibung Diese Funktion setzt die aktuelle Hintergrundfarbe auf die durch den Parameter *Color* vorgegebene Farbe oder aber auf die

ähnlichste physikalische Farbe, wenn das Gerät diesen RGB-Farbwert nicht darstellen kann.

Wenn der Hintergrundmodus *OPAQUE* ist, verwendet die GDI die Hintergrundfarbe, um Lücken zwischen Linien bestimmter Arten, Lücken zwischen schraffierten Linien in Pinseln und zwischen Zeichenzellen auszufüllen. Die GDI verwendet die Hintergrundfarbe auch dazu, Bitmaps aus der Farbdarstellung in monochrome Darstellung und umgekehrt zu konvertieren.

Der Hintergrundmodus wird durch die Funktion *SetBkMode* gesetzt. Siehe auch *BitBlt* und *StretchBlt*.

DC: Bezeichnet den Gerätekontext.

Color: Bestimmt die neue Hintergrundfarbe.

Rückgabewert Der Rückgabewert gibt die vorherige Hintergrundfarbe als einen RGB-Farbwert an. Tritt ein Fehler auf, ist der Rückgabewert 0x80000000.

Querverweis *BitBlt, SetBkMode, StretchBlt*

SetBkMode

Syntax
```
int SetBkMode(HDC DC, int BkMode);
```

Beschreibung Diese Funktion setzt den Hintergrundmodus, der für Text und Linien verwendet wird. Der Hintergrundmodus bestimmt, ob die GDI vorhandene Hintergrundfarben auf der Bildschirmfläche löschen soll, bevor Text, Pinselschraffur oder irgendein Stift, der keine durchgezogene Linie erzeugt, dargestellt wird.

DC: Bezeichnet den Gerätekontext.

BkMode: Bestimmt den Hintergrundmodus. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
OPAQUE	Hintergrund wird mit der aktuellen Hintergrundfarbe ausgefüllt, bevor gezeichnet wird.
TRANSPARENT	Hintergrund bleibt unberührt.

Rückgabewert Der Rückgabewert gibt den vorherigen Hintergrundmodus an.

SetBrushOrg

Syntax
```
DWORD SetBrushOrg(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion setzt den Ursprung des aktuell ausgewählten Pinsels im gegebenen Gerätekontext.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die x-Koordinate des neuen Ursprungs in gerätespezifischen Einheiten. Der Wert muß im Bereich von 0 bis 7 liegen.

Y: Bestimmt die y-Koordinate des neuen Ursprungs in gerätespezifischen Einheiten. Der Wert muß im Bereich von 0 bis 7 liegen.

Rückgabewert Der Rückgabewert bestimmt den vorherigen Ursprung des Pinsels. Die vorherige x-Koordinate liegt im niederwertigen, die y-Koordinate im höherwertigen WORD.

Hinweis Die Ausgangskoordinaten des Pinselursprungs sind (0,0). Die Funktion *SetBrushOrg* sollte nicht für vordefinierte Objekte verwendet werden.

SetCapture

Syntax
```
HWND SetCapture(HWND Wnd);
```

Beschreibung Diese Funktion bewirkt, daß alle nachfolgenden Mauseingaben an das durch den Parameter *Wnd* bezeichnete Fenster gesendet werden, unabhängig von der Mauszeigerposition.

Wnd: Bezeichnet das Fenster, das alle Mauseingaben abfangen soll.

Rückgabewert Der Rückgabewert gibt das Fenster an, das vorher alle Mauseingaben erhalten hatte. Er ist 0, wenn keines existiert.

Hinweis Wenn das Fenster keine exkluxive Benutzung der Maus mehr benötigt, dann sollte die Funktion *ReleaseCapture* aufgerufen werden, damit andere Fenster die Maus exklusiv nutzen können.

Querverweis *ReleaseCapture*

SetCaretBlinkTime

Syntax
```
void SetCaretBlinkTime(WORD MSeconds);
```

Beschreibung Diese Funktion setzt die Blinkrate des Carets (die Zeit zwischen dem Aufblinken des Carets) auf die durch den Parameter *MSeconds* angegebene Zahl von Millisekunden.

Das Caret blinkt auf oder erlischt alle *MSeconds* Millisekunden. Das bedeutet, daß ein kompletter Blinkvorgang (AN-AUS-AN) 2 * *MSeconds* Millisekunden beansprucht.

MSeconds: Bestimmt die neue Blinkrate in Millisekunden.

Rückgabewert Keiner.

Hinweis Das Caret ist eine gemeinsam genutzte Ressource. Ein Fenster sollte die Blinkrate des Carets nur setzen, wenn es das Caret besitzt. Verliert das Fenster den Fokus oder wird es inaktiv, sollte das vorherige Caret wiederhergestellt werden.

SetCaretPos

Syntax `void SetCaretPos(int X, int Y);`

Beschreibung Diese Funktion bewegt das Caret zu der Position, die in virtuellen Koordinaten durch die Parameter *X* und *Y* angegeben ist.

Virtuelle Koordinaten sind relativ zum Client-Bereich eines Fensters gebildet und werden vom Abbildungsmodus des Fensters beeinflußt, so daß die exakte Position in Pixeln von diesem Modus abhängt. *SetCaretPos* bewegt das Caret nur, wenn es von einem Fenster der aktuellen Task besessen wird, unabhängig davon, ob das Caret ausgeblendet ist oder nicht.

X: Bestimmt die neue x-Koordinate des Carets in virtuellen Koordinaten.

Y: Bestimmt die neue y-Koordinate des Carets in virtuellen Koordinaten.

Rückgabewert Keiner.

Hinweis Das Caret ist eine gemeinsam genutzte Ressource. Ein Fenster sollte das Caret jeweils nur dann bewegen, wenn es das Caret besitzt.

SetClassLong

Syntax `LONG SetClassLong(HWND Wnd, int Index, LONG NewLong);`

Beschreibung Diese Funktion ersetzt den vom Parameter *Index* bezeichneten LONG-Wert in der *WNDCLASS*-Datenstruktur des durch den Parameter *Wnd* bezeichneten Fensters.

Wnd: Bezeichnet das Fenster.

Index: Bestimmt den Byte-Offset des LONG-Wertes, der ersetzt werden soll. Dieser Parameter kann auch einen der folgenden Werte annehmen:

Wert	Bedeutung
GCL_MENUNAME	Setzt einen neuen Zeiger auf die Menübezeichnung.
GCL_WNDPROC	Setzt einen neuen Zeiger auf die Fensterfunktion.

NewLong: Bestimmt den Ersatzwert.

Rückgabewert Der Rückgabewert zeigt den vorherigen Wert des bezeichneten LONG-Wertes an.

Hinweis Wenn die Funktion *SetClassLong* und der Index *GCL_WNDPROC* verwendet werden, um eine Fensterfunktion zu setzen, muß die angegebene Funktion Fensterfunktionsform haben und in der Moduldefinitionsdatei exportiert werden. Beachte hierzu die Beschreibung Funktion *RegisterClass*.

Der Aufruf von *SetClassLong* mit dem Index *GCL_WNDPROC* erzeugt eine neue Unterklasse der Fensterklasse, die alle anschließend mit dieser Klasse erzeugten Fenster betrifft. Eine Anwendung sollte keine Fensterunterklassen für Standard-Windows-Steuerelemente (z.B. Kombinationsfenster und Schalter) erstellen.

Um auf zusätzliche Zwei-Byte-Werte zuzugreifen, die belegt wurden, als die Fensterklasse erzeugt wurde, wird ein positiver Byte-Offset ab dem vom Parameter *Index* bezeichneten Index verwendet.Dieser Index beginnt bei 0 für den ersten Zwei-Byte-Wert im zusätzlichen Speicherbereich, 2 für den nächsten Zwei-Byte-Wert und so weiter.

SetClassWord

Syntax
```
WORD SetClassWord(HWND Wnd, int Index, WORD NewWord);
```

Beschreibung Diese Funktion ersetzt das vom Parameter *Index* bezeichnete WORD in der *WNDCLASS*-Datenstruktur des durch den Parameter *Wnd* bezeichneten Fensters.

Wnd: Bezeichnet das Fenster.

Index: Bestimmt den Byte-Offset des Wortes, das verändert werden soll. Dieser Parameter kann einen der folgenden Werte annehmen:

Wert	Bedeutung
GCW_CBCLSEXTRA	Setzt zwei neue Bytes zusätzlicher Fensterklassendaten.
GCW_CBWNDEXTRA	Setzt zwei neue Bytes zusätzlicher Fensterklassendaten.
GCW_HBRBACKGROUND	Setzt ein neues Handle auf einen Hintergrundpinsel.
GCW_HCURSOR	Setzt ein neues Handle auf einen Mauszeiger.
GCW_HICON	Setzt ein neues Handle auf ein Symbol.
GCW_STYLE	Setzt ein neues Stilbit für die Fensterklasse.

NewWord: Bestimmt den Ersatzwert.

Rückgabewert Der Rückgabewert gibt den vorherigen Wert des bezeichneten Wortes an.

Hinweis Die Funktion *SetClassWord* sollte mit Vorsicht angewandt werden. So ist es zum Beispiel möglich, die Hintergrundfarbe für eine Klasse mit *SetClassWord* zu ändern. Diese Änderung bewirkt aber nicht, daß alle Fenster, die dieser Klasse angehören, sofort neu gezeichnet werden.

Um auf zusätzliche Vier-Byte-Werte zuzugreifen, die belegt wurden, als die Fensterklasse erzeugt wurde, wird ein positiver Byte-Offset ab dem vom Parameter *Index* bezeichneten Index verwendet. Dieser Index beginnt bei 0 für den ersten Vier-Byte-Wert im zusätzlichen Speicherbereich, 4 für den nächsten Vier-Byte-Wert und so weiter.

SetClipboardData

Syntax
```
HANDLE SetClipboardData(WORD Format, HANDLE Mem);
```

Beschreibung Diese Funktion setzt ein Daten-Handle zur Zwischenablage für die die durch den Parameter *Mem* bezeichneten Daten. Es wird vorausgesetzt, daß die Daten das im Parameter *Format* vorgegebene Format haben. Nach dem Setzen eines Zwischenablagedaten-Handle gibt die Funktion *SetClipboardData* den von *Mem* bezeichneten Speicherblock frei.

Format: Bestimmt ein Datenformat. Es kann eines der im »Hinweis« beschriebenen Formate sein. Zusätzlich zu den vordefinierten Formaten kann jeder mit der Funktion *RegisterClipboardFormat* registrierte Formatwert als Parameter *Format* verwendet werden.

Mem: Bezeichnet den globalen Speicherblock, der die Daten im angegebenen Format enthält. Der Parameter *Mem* kann auch 0 sein. In diesem Fall muß die Anwendung die Daten nicht formatieren und kein Handle auf sie bereitstellen, solange sie nicht durch eine Botschaft *WM_RENDERFORMAT* dazu aufgefordert wird.

Rückgabewert Der Rückgabewert bezeichnet die Daten und wird von der Zwischenablage zugewiesen.

Hinweis Sobald der Parameter *Mem* an *SetClipboardData* übergeben wurde, wird der bezeichnete Datenblock Eigentum der Zwischenablage. Die Anwendung kann die Daten zwar lesen, sollte aber den Block nicht freigeben oder gesperrt lassen.

Windows unterstützt zwei Formate für Text, *CF_TEXT* und *CF_OEMTEXT*. *CF_TEXT* ist das voreingestellte Zwischenablageformat für Text aus Windows-Anwendungen. Das Format *CF_OEMTEXT* wird für Text aus Nicht-Windows-Anwendungen verwendet.

Wird *GetClipboardData* aufgerufen, um Daten in einem Textformat zu liefern und das andere Textformat ist dabei das einzig verfügbare, konvertiert Windows den Text automatisch in das erforderliche Format, bevor er der Anwendung übergeben wird.

Eine Anwendung registriert andere Standardformate wie Rich Text Format (RTF) durch Benennung über die Funktion *RegisterClipboardFormat* anstatt mit einer symbolischen Konstante.

Wert	Bedeutung
CF_BITMAP	Ein Handle auf ein Bitmap (*HBITMAP*).
CF_DIB	Ein Speicherblock, der die *BITMAPINFO*-Datenstruktur gefolgt von den Bitmap-Bits enthält.
CF_DIF	Software Arts' Data Interchange Format.
CF_DSPBITMAP	Bitmap-Anzeigeformat, das einem privaten Format zugeordnet ist. Der Parameter *Mem* muß ein Handle auf Daten sein, die im Bitmap-Format anstelle eines privaten Formats dargestellt werden können.
CF_DSPMETAFILEPICT	Metafile-Darstellungsformat, das einem privaten Format zugeordnet ist. Der Parameter *Mem* muß ein Handle auf Daten sein, die im Metafile-Bildformat statt einem privaten Format dargestellt werden können.
CF_DSPTEXT	Textdarstellungsformat, das einem privaten Format zugeordnet ist. Der Parameter *Mem* muß ein Handle auf Daten sein, die im Textformat anstelle eines privaten Formats dargestellt werden können.
CF_METAFILEPICT	Metafile-Bildformat wie von der Metafile-Pict-Datenstruktur definiert.
CF_OEMTEXT	Textformat, das Zeichen aus dem OEM-Zeichensatz enthält. Jede Zeile endet mit einer Kombination aus Wagenrücklauf/Zeilenschaltung (CR-LF). Ein *Null*-Zeichen kennzeichnet das Datenende.
CF_OWNERDISPLAY	Besitzeranzeigeformat. Der Besitzer der Zwischenablage muß das Anwendungsfenster der Zwischenablage anzeigen und aktualisieren und empfängt die Botschaften *WM_ASKCBFORMATNAME*, *WM_HSCROLLCLIPBOARD*, *WM_PAINTCLIPBOARD*, *WM_SIZECLIPBOARD* und *WM_VSCROLLCLIPBOARD*. Der Parameter *Mem* muß 0 sein.

Wert	Bedeutung
CF_PALETTE	Handle auf eine Farbpalette. Speichert eine Anwendung Daten in der Zwischenablage, die von einer Farbpalette abhängen oder eine solche voraussetzen, sollte sie die Palette ebenfalls in der Zwischenablage speichern. Wenn die Zwischenablage Daten im Format *CF_PALETTE* (virtuelle Farbpalette) enthält, sollte die Anwendung davon ausgehen, daß andere Daten in der Zwischenablage entgegen dieser virtuellen Palette realisiert sind. Die Viewer-Anwendung der Zwischenablage (CLIPBRD.EXE) benutzt als aktuelle Palette ein Objekt im Format *CF_PALETTE*, das in der Zwischenablage ist, wenn die anderen Formate in der Zwischenablage dargestellt werden.
CF_PRIVATEFIRST bis *CF_PRIVATELAST*	Bereich von Integerwerten, die für private Formate benutzt werden. Daten-Handle, die Formaten in diesem Bereich zugeordnet sind, werden nicht automatisch freigegeben. Alle Daten-Handle müssen freigegeben werden, bevor die Anwendung beendet oder wenn eine Botschaft *WM_DESTROYCLIPBOARD* empfangen wird.
CF_SYLK	Microsoft-Symbolic-Link (SYLK) Format.
CF_TEXT	Textformat. Jede Zeile endet mit einer Kombination Wagenrücklauf/Zeilenschaltung. Ein *Null*-Zeichen kennzeichnet das Datenende.
CF_TIFF	Tag Image File Format.

SetClipboardViewer

Syntax

```
HWND SetClipboardViewer(HWND Wnd);
```

Beschreibung

Diese Funktion fügt das durch den Parameter *Wnd* bezeichnete Fenster an die Kette von Fenstern an, die über die Botschaft *WM_DRAWCLIPBOARD* benachrichtigt werden, sobald der Inhalt der Zwischenablage verändert wird.

Wnd: Bezeichnet das Fenster, das Botschaften an die Zwischenablage-Viewer-Kette empfangen soll.

Rückgabewert Der Rückgabewert bezeichnet das nächste Fenster in der Kette der Zwischenablage-Viewer.

Dieses Handle sollte im statischen Speicher abgelegt und zur Beantwortung der Botschaften an die Kette der Zwischenablage-Viewer verwendet werden.

Hinweis Fenster, die Teil der Zwischenablage-Viewer-Kette sind, müssen auf die Botschaften *WM_CHANGECBCHAIN, WM_DRAWCLIPBOARD* und *WM_DESTROY* antworten.

Wenn eine Anwendung sich selbst aus der Zwischenablage-Viewer-Kette löschen will, muß sie die Funktion *ChangeClipboardChain* aufrufen.

SetCommBreak

Syntax
```
int SetCommBreak(int Cid);
```

Beschreibung Diese Funktion unterbricht die Zeichenübertragung und versetzt die Übertragungsleitung in einen Wartezustand, bis die Funktion *ClearCommBreak* aufgerufen wird.

Cid: Bestimmt die Schnittstelle, die unterbrochen werden soll. Die Funktion *OpenComm* gibt diesen Wert zurück.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war und negativ, wenn *Cid* keine gültige Schnittstelle angibt.

Querverweis *ClearCommBreak, OpenComm*

SetCommEventMask

Syntax
```
WORD SetCommEventMask(int Cid, WORD EvtMask);
```

Beschreibung Diese Funktion aktiviert und liefert die Ereignismaske der vom Parameter *Cid* angegebenen Schnittstelle.

Die Bits des Parameters *EvtMask* definieren, welche Ereignisse aktiviert werden sollen. Der Rückgabewert zeigt auf den aktuellen Status der Ereignismaske.

Cid: Bestimmt die Schnittstelle, die aktiviert werden soll. Die Funktion *OpenComm* gibt diesen Wert zurück.

EvtMask: Bestimmt, welche Ereignisse aktiviert werden sollen. Er kann jede Kombination der im Hinweis aufgeführten Werte sein.

Rückgabewert Der Rückgabewert zeigt auf eine Word-Ereignismaske. Jedes Bit in der Ereignismaske gibt an, ob ein bestimmtes Ereignis eingetreten ist. In diesem Fall ist das entsprechende Bit gesetzt (1). Die möglich:en Ergebnisse sind in der folgenden Tabelle beschrieben.

Wert	Bedeutung
EV_BREAK	Gesetzt, wenn eine Unterbrechung bei der Eingabe erkannt wurde.
EV_CTS	Gesetzt, wenn das Signal Clear-To-Send (*CTS*) den Status wechselt.
EV_DSR	Gesetzt, wenn das Signal Data-Set-Ready (*DSR*) den Status wechselt.
EV_ERR	Gesetzt, wenn ein Leitungsstatusfehler auftritt *CE_FRAME, CE_OVERRUN* oder *CE_RXPARITY*)..
EV_PERR	Gesetzt, wenn ein Druckerfehler an einer parallelen Schnittstelle erkannt wurde. Fehler sind *CE_DNS, CE_IOE, CE_LOOP* und *CE_PTO.*
EV_RING	Gesetzt, wenn ein Anrufsignal erkannt wurde.
EV_RLSD	Gesetzt, wenn das Signal Receive-Line-Signal-Detect (*RLSD*) den Status ändert.
EV_RXCHAR	Gesetzt, wenn ein Zeichen empfangen und in die Warteschlange gestellt wurde.
EV_RXFLAG	Gesetzt, wenn das Ereigniszeichen empfangen und in die Empfangs-Warteschlange gestellt wurde. Das Ereigniszeichen wird im Gerätesteuerblock des Gerätes bestimmt.
EV_TXEMPTY	Gesetzt, wenn das letzte Zeichen in der Sendewarteschlange gesendet wurde.

SetCommState

Syntax `int SetCommState(DCB FAR * DCB);`

Beschreibung Diese Funktion versetzt eine Schnittstelle in den Status, den der Gerätesteuerblock in der *DCB*-Struktur des Parameters *DCB* vorgibt. Das zu setzende Gerät muß durch das ID-Feld des Steuerblocks bezeichnet werden. Sie initialisiert die gesamte Hardware und die Steuerleitungen gemäß der Definition in *DCB*. Sie leert keine Sende- oder Empfangswarteschlangen.

DCB: Eine *DCB*-Datenstruktur, die die gewünschten Schnittstelleneinstellungen für das Gerät enthält.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war. Er ist negativ, wenn ein Fehler auftritt.

SetCursor

Syntax `HCURSOR SetCursor(HCURSOR Cursor);`

Beschreibung Diese Funktion setzt die Mauszeigerform auf die durch den Parameter *Cursor* bezeichnete Form. Der Mauszeiger wird nur

gesetzt, wenn sich die neue Form von der aktuellen Form unterscheidet, andernfalls kehrt die Funktion sofort zurück.

Ist *Cursor* 0, wird der Mauszeiger vom Bildschirm gelöscht.

Cursor: Bezeichnet die Mauszeigerressource. Die Ressource muß vorher mit der Funktion *LoadCursor* geladen worden sein.

Rückgabewert Der Rückgabewert bezeichnet die Mauszeiger-Ressource, die die vorherige Mauszeigerform definiert. Er ist 0, wenn es keine vorherige Mauszeigerform gibt.

Hinweis Der Mauszeiger ist eine gemeinsam genutzte Ressource. Eine Anwendung sollte den Mauszeiger nur setzen, wenn sich der Mauszeiger in ihrem Client-Bereich befindet oder sie den Mausfang hat. In Systemen ohne Maus sollte das Fenster die vorherige Mauszeigerform wiederherstellen, bevor der Mauszeiger den Client-Bereich verläßt oder bevor das Fenster einem anderen Fenster die Steuerung überläßt.
Jede Anwendung, die die Mauszeigerform ändern muß, während sich dieser in einem ihrer Fenster befindet, muß sicherstellen, daß der Klassenzeiger für die angegebene Klasse des Fensters auf 0 gesetzt ist. Ist der Klassenzeiger nicht 0, stellt Windows jedesmal, wenn die Maus bewegt wird, die vorherige Form wieder her.
Der Mauszeiger ist auf dem Bildschirm nicht sichtbar, wenn sein Anzeigezähler durch entsprechende Aufrufe von *ShowCursor* auf 0 herabgesetzt wurde.

SetCursorPos

Syntax
```
void SetCursorPos(int X, int Y);
```

Beschreibung Diese Funktion bewegt den Mauszeiger zu den durch die Parameter *X* und *Y* angegebenen Bildschirmkoordinaten. Wenn die neuen Koordinaten nicht innerhalb des Bildschirmrechtecks liegen, das durch den letzten Aufruf der Funktion *ClipCursor* gesetzt wurde, berichtigt Windows die Koordinaten automatisch so, daß der Mauszeiger innerhalb des Rechtecks bleibt.

X: Bestimmt die neue x-Koordinate des Mauszeigers in Bildschirmkoordinaten.

Y: Bestimmt die neue y-Koordinate des Mauszeigers in Bildschirmkoordinaten.

Rückgabewert Keiner.

Hinweis Der Mauszeiger ist eine gemeinsam genutzte Ressource. Ein Fenster sollte den Mauszeiger nur bewegen, wenn er sich in seinem Client-Bereich befindet.

SetDIBits

Syntax

```
int SetDIBits(HDC DC, HANDLE Bitmap, WORD StartScan,
WORD NumScans, LPSTR Bits, LPBITMAPINFO BitsInfo,
WORD Usage);
```

Beschreibung

Diese Funktion setzt die Bits eines Bitmaps auf die in einer geräteunabhängigen Bitmapbeschreibung (DIB) angegebenen Werte.

DC: Bezeichnet den Gerätekontext.

Bitmap: Bezeichnet das Bitmap.

StartScan: Bestimmt die Nummer der ersten Scan-Zeile in *Bits*.

NumScans: Bestimmt die Anzahl von Scan-Zeilen in *Bits* und die Anzahl von Zeilen, die in dem vom Parameter *Bitmap* bezeichneten Bitmap gesetzt werden sollen.

Bits: Zeigt auf die geräteunabhängigen Bitmapbits, die als ein Array von Bytes gespeichert sind. Das Format der Bitmap-Werte hängt vom Feld *biBitCount* der durch BitsInfo bezeichneten *BITMAPINFO*-Datenstruktur ab.

BitsInfo: Zeigt auf eine *BITMAPINFO*-Datenstruktur, die Informationen über das geräteunabhängige Bitmap enthält.

Usage: Bestimmt, ob die Felder *bmiColors*[*i*] in *BitsInfo* explizite RGB-Werte oder Indizes in die aktuell realisierte virtuelle Palette enthalten.

Der Parameter *Usage* muß einen der folgenden Werte annehmen:

Wert	Bedeutung
DIB_PAL_COLORS:	Die Farbtabelle besteht aus einem Array von 16-Bit-Indizes in die aktuell realisierte virtuelle Palette.
DIB_RGB_COLORS:	Die Farbtabelle enthält explizite RGB-Werte.

Rückgabewert

Bei erfolgreicher Ausführung wird die Anzahl von Scan-Zeilen, die erfolgreich kopiert wurden, zurückgeliefert, andernfalls 0.

Hinweis

Das Bitmap, das vom Parameter *Bitmap* bezeichnet wird, darf nicht für einen Gerätekontext gewählt sein, wenn die Anwendung diese Funktion aufruft.

Der Ursprung für geräteunabhängige Bitmaps ist die untere linke Ecke des Bitmaps. Die obere linke Ecke ist der Ursprung, wenn der Abbildungsmodus *MM_TEXT* ist.

SetDIBitsToDevice

Syntax

```
int SetDIBitsToDevice(HDC DC, WORD DestX, WORD DestY,
WORD Width, WORD Height, WORD SrcX, WORD SrcY,
WORD StartScan, WORD NumScans, LPSTR Bits,
LPBITMAPINFO BitsInfo, WORD Usage);
```

Beschreibung

Diese Funktion kopiert Bits aus einem geräteunabhängigen Bitmap (DIB) direkt auf eine Bildschirmoberfläche. Die Parameter *SrcX*, *SrcY*, *Width* und *Height* definieren ein Rechteck innerhalb des gesamten DIB. *SetDIBitsToDevice* setzt die Bits in diesem Rechteck direkt auf der Bildschirmoberfläche des durch den Parameter *DC* bezeichneten Ausgabegerätes an die durch die Parameter *DestX* und *DestY* festgelegte Position.

Um den erforderlichen Speicherbedarf beim Setzen von Bits aus einem großen DIB auf eine Bildschirmfläche zu verringern, kann eine Anwendung die Ausgabe durch wiederholtes Aufrufen von *SetDIBitsToDevice* eingrenzen, indem sie dabei jedes Mal einen anderen Teil des gesamten DIB in *Bits* bearbeitet.

Die Werte der beiden Parameter *StartScan* und *NumScan* bezeichnen den Teil des gesamten DIB, der im Puffer *Bits* enthalten ist.

DC: Bezeichnet den Gerätekontext.

DestX: Bestimmt die x-Koordinate des Ursprungs des Zielrechtecks.

DestY: Bestimmt die y-Koordinate des Ursprungs des Zielrechtecks.

Width: Bestimmt die x-Abmessung des Rechtecks im DIB.

Height: Bestimmt die y-Abmessung des Rechtecks im DIB.

SrcX: Bestimmt die x-Koordinate der Quelle im DIB.

SrcY: Bestimmt die y-Koordinate der Quelle im DIB.

StartScan: Bestimmt die Nummer der Scan-Zeilen des DIB, die in der ersten Scan-Zeile des Puffers in *Bits* enthalten ist.

NumScans: Bestimmt die Anzahl von Scan-Zeilen des DIB, die in *Bits* enthalten sind.

Bits: Zeigt auf ein Byte-Array, das die Bits des geräteunabhängigen Bitmaps enthält. Das Format der Bits ist durch das Feld *biBitCount* in *BitsInfo* festgelegt.

BitsInfo: Zeigt auf eine *BITMAPINFO*-Datenstruktur, die Informationen über das DIB enthält.

Usage: Bestimmt, ob die Felder *bmiColors*[*i*] in *BitsInfo* explizite RGB-Werte oder Indizes in die aktuell realisierte virtuelle Pa-

lette enthalten. Der Parameter *Usage* muß einen der folgenden Werte annehmen:

Wert	Bedeutung
DIB_PAL_COLORS:	Die Farbpalette besteht aus einem Array von 16-Bit-Indizes in die aktuell realisierte virtuelle Palette.
DIB_RGB_COLORS:	Die Farbtabelle enthält explizite RGB-Werte.

Rückgabewert Bei erfolgreicher Ausführung ist der Rückgabewert die Anzahl der gesetzten Scan-Zeilen.

Hinweis Alle Koordinaten außer den virtuellen Koordinaten *DestX* und *DestY* sind gerätebezogene Koordinaten (also die Koordinaten des DIB).

SetDlgItemInt

Syntax
```
void SetDlgItemInt(HWND Dlg, int IDDlgItem, WORD Value,
BOOL Signed);
```

Beschreibung Diese Funktion setzt den Text eines Steuerelements in dem angegebenen Dialogfenster auf den String, den der vom Parameter *Value* gegebene Integerwert bestimmt. Die Funktion *SetDlgItemInt* wandelt *Value* in einen aus Dezimalziffern bestehenden String um und kopiert den String dann in das Steuerelement. Wenn der Parameter *Signed* ungleich Null ist, wird angenommen, daß *Value* vorzeichenbehaftet ist. Ist *Value* vorzeichenbehaftet und kleiner 0, setzt die Funktion ein Minuszeichen vor die erste Ziffer im String. *SetDlgItemInt* sendet eine Botschaft *WM_SETTEXT* an das angegebene Steuerelement.

Dlg: Bezeichnet das Dialogfenster, das das Steuerelement enthält.

IDDlgItem: Bestimmt das Steuerelement, das verändert werden soll.

Value: Bestimmt den Wert, der gesetzt werden soll.

Signed: Bestimmt, daß der Wert vorzeichenbehaftet ist, wenn Signed einen Wert ungleich Null hat.

Querverweis *WM_SETTEXT*

SetDlgItemText

Syntax
```
void SetDlgItemText(HWND Dlg, int IDDlgItem, LPSTR Str);
```

Beschreibung Diese Funktion setzt die Überschrift oder den Text eines Steuerelements in das durch den Parameter *Dlg* bezeichnete Dia-

logfenster. Sie sendet eine Botschaft *WM_SETTEXT* an das angegebene Steuerelement.

Dlg: Bezeichnet das Dialogfenster, welches das Steuerelement enthält.

IDDlgItem: Bestimmt das Steuerelement, dessen Text gesetzt werden soll.

Str: Zeigt auf einen nullterminierten String, der in das Steuerelement kopiert werden soll.

Rückgabewert Keiner.

Querverweis *WM_SETTEXT*

SetDoubleClickTime

Syntax
```
void SetDoubleClickTime(WORD Count);
```

Beschreibung Diese Funktion setzt das Doppelklickzeitintervall für die Maus.

Ein Doppelklick ist eine Folge von zwei Klicks mit der Maustaste, wobei der zweite innerhalb einer festgelegten Zeitspanne nach dem ersten erfolgt. Das Doppelklickzeitintervall ist die maximale Zahl von Millisekunden, die zwischen dem ersten und zweiten Klick eines Doppelklicks verstreichen dürfen.

Count: Bestimmt die Zahl der Millisekunden, die zwischen den beiden Klicks liegen dürfen.

Rückgabewert Keiner.

Hinweis Ist der Parameter *Count* auf 0 gesetzt, verwendet Windows das Standardzeitintervall von 500 Millisekunden für den Doppelklick.

Die Funktion *SetDoubleClickTime* wechselt das Doppelklickzeitintervall für alle Fenster im System.

SetEnvironment

Syntax
```
int SetEnvironment(LPSTR PortName, LPSTR Environ,
WORD Count);
```

Beschreibung Diese Funktion kopiert den Inhalt des durch den Parameter *Environ* bezeichneten Puffers in die Umgebung, die dem Gerät zugeordnet ist, das mit dem vom Parameter *PortName* bezeichneten Anschluß verbunden ist. *SetEnvironment* löscht jede existierende Umgebung. Gibt es keine Umgebung für den angegebenen Anschluß, wird diese von *SetEnvironment* erzeugt. Ist der Parameter *Count* 0, wird die vorhandene Umgebung nur gelöscht und nicht ersetzt.

PortName: Zeigt auf einen nullterminierten String, der die Bezeichnung des gewünschten Anschlusses enthält.

Environ: Zeigt auf einen Puffer, der die neue Umgebung enthält.

Count: Bestimmt die Zahl der Bytes, die kopiert werden sollen.

Rückgabewert Bei erfolgreicher Ausführung wird die Anzahl der Bytes zurückgegeben, die tatsächlich in die Umgebung kopiert wurden. Der Rückgabewert ist 0, wenn ein Fehler auftritt, und -1, wenn die Umgebung gelöscht ist.

Hinweis Das erste Feld im Puffer *Environ* muß das mit dem Parameter *DeviceName* der Funktion *CreateDC* übergebene sein. Bezeichnet *PortName* einen Nullanschluß (wie in der Datei WIN.INI vereinbart), wird die Gerätebezeichnung, auf die *Environ* zeigt, verwendet, um die gewünschte Umgebung anzulegen.

SetErrorMode

Syntax
```
BOOL SetErrorMode(WORD Mode);
```

Beschreibung Diese Funktion steuert, ob Windows Fehler des DOS-Interrupt 24H (schwere I/O-Fehler, die ohne Korrektur zum Programmabbruch führen würden) übernimmt oder ob dies der aufrufenden Anwendung überlassen wird. Windows fängt alle Fehler des INT 24H ab. Dabei gibt es zwei Möglichkeiten:

- Wenn die Anwendung *SetErrorMode* mit auf 0 gesetztem Parameter *Mode* aufruft und anschließend ein Fehler des INT 24H auftritt, stellt Windows ein Fehlermeldungsfenster dar.

- Wenn die Anwendung *SetErrorMode* mit auf 1 gesetztem Parameter *Mode* aufruft und ein Fehler des INT 24H auftritt, stellt Windows das Standard-Fehlermeldungsfenster für den DOS-Interrupt 24H nicht dar, sondern gibt den originalen Fehlerrückruf des INT 21H an die Anwendung weiter. Dies ermöglicht es der Anwendung, Diskettenfehler unter Verwendung von INT 21H, AH=59H (Erweiterten Fehlercode holen) geeignet zu bearbeiten.

Mode: Bestimmt das Fehlermodus-Flag. Ist *Mode* auf 0 gesetzt, stellt Windows ein Fehlermeldungsfenster dar, wenn ein Fehler INT 24H auftritt. Ist *Mode* auf 1 gesetzt, gibt Windows den Fehlerrückruf des INT 21H an die aufrufende Anwendung weiter und stellt kein Meldungsfenster dar.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn *Mode* auf 1 gesetzt ist, andernfalls Null.

SetFocus

Syntax
```
HWND SetFocus(HWND Wnd);
```

Beschreibung Diese Funktion weist dem vom Parameter *Wnd* bezeichneten Fenster den Fokus zu. Der Fokus leitet alle nachfolgenden Tastatureingaben an das angegebene Fenster. Das Fenster, das vorher den Fokus hatte, verliert ihn. Ist *Wnd* 0, werden Tasteneingaben ignoriert. *SetFocus* sendet eine Botschaft *WM_KILLFOCUS* an das Fenster, das den Fokus verliert und eine Botschaft *WM_SETFOCUS* an das Fenster, das den Fokus erhält. Sie aktiviert auch entweder das Fenster, das den Fokus empfängt, oder dessen übergeordnetes Fenster.

Wnd: Bezeichnet das Fenster, das den Fokus erhalten soll.

Rückgabewert Der Rückgabewert bezeichnet das Fenster, das vorher den Fokus hatte. Er ist 0, wenn kein solches Fenster existiert.

Hinweis Wenn ein Fenster aktiv ist, aber den Fokus nicht hat (das bedeutet, daß kein Fenster den Fokus hat), produziert jeder Tastendruck eine Botschaft *WM_SYSCHAR, WM_SYSKEYDOWN* oder *WM_SYSKEYUP.*

Ist die Taste *VK_MENU* ebenfalls gedrückt, ist das Bit 30 im Parameter *lParam* der Botschaft gesetzt, andernfalls ist dieses Bit in den erzeugten Botschaften nicht gesetzt.

SetHandleCount

Syntax
```
WORD SetHandleCount(WORD Number);
```

Beschreibung Diese Funktion ändert die Anzahl der für eine Task verfügbaren Datei-Handle. Standardmäßig ist die Zahl der für eine Task verfügbaren Datei-Handle 20.

Number: Bestimmt die Zahl der Datei-Handle, die von der Task verwendet werden können. Das Maximum ist 255.

Rückgabewert Es wird die Zahl der tatsächlich verfügbaren Datei-Handle für die Task zurückgeliefert. Diese kann kleiner sein als die Zahl, die vom Parameter *Number* vorgegeben ist.

SetKeyboardState

Syntax
```
void SetKeyboardState(BYTE FAR * KeyState);
```

Beschreibung Diese Funktion kopiert die 256 Bytes im Parameter *KeyState* in die Tastaturstatus-Tabelle von Windows.

KeyState: Ein Array von 256 Bytes, das den Status der einzelnen Tasten enthält.

Hinweis Eine Anwendung sollte zuerst die Funktion *GetKeyboardState* aufrufen, um das 256-Byte-Array zu initialisieren, und erst danach die gewünschten Bytes ändern.

Die Funktion *SetKeyboardState* setzt die LEDs und die BIOS-Flags für die Tasten NumLock, CapsLock und ScrollLock entsprechend des Arretierungszustandes der Einträge *VK_NUMLOCK, VK_CAPITAL* und *VK_OEM_SCROLL* im Array.

Rückgabewert Keiner.

Querverweis *GetKeyboardState*

SetMapMode

Syntax
```
int SetMapMode(HDC DC, int MapMode);
```

Beschreibung Diese Funktion setzt den Abbildungsmodus des bezeichneten Gerätekontextes. Der Abbildungsmodus definiert die Maßeinheit, die bei der Übersetzung virtueller Einheiten in gerätebezogene Einheiten benutzt wird und bestimmt die Ausrichtung der x- und y-Achsen des Geräts. Die GDI benutzt den Abbildungsmodus, um virtuelle Koordinaten in passende gerätebezogene Koordinaten umzuwandeln.

DC: Bezeichnet den Gerätekontext.

MapMode: Bestimmt den neuen Abbildungsmodus. Er kann einer der in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »MM_XXXX« beschriebenen Werte sein.

Rückgabewert Es wird der vorherigen Abbildungsmodus zurückgeliefert.

Hinweis Der Modus *MM_TEXT* erlaubt es Anwendungen, geräteeigene Pixel zu verwenden, deren Größe von Gerät zu Gerät variiert.

Die Modi *MM_HIENGLISH, MM_HIMETRIC, MM_LOENGLISH, MM_LOMETRIC* und *MM_TWIPS* sind nützlich für Anwendungen, die in physikalischen Einheiten zeichnen müssen (wie zum Beispiel in Zoll oder Millimetern).

Der Modus *MM_ISOTROPIC* stellt ein Seitenverhältnis 1:1 her. Er wird dann verwendet, wenn die exakte Form eines Bildes erhalten werden soll.

Der Modus *MM_ANISOTROPIC* erlaubt es, die x- und y-Koordinaten unabhängig voneinander einzustellen.

SetMapperFlags

Syntax
```
DWORD SetMapperFlags(HDC DC, DWORD Flag);
```

Beschreibung Diese Funktion wechselt den Algorithmus, den der Schriftenabbilder verwendet, um virtuelle Schriften auf physikalische Schriften abzubilden. Ist das erste Bit des Parameters *Flag* auf

1 gesetzt, wählt der Abbilder nur Schriften, deren x-und y-Abmessungen genau mit denen des angegebenen Geräts übereinstimmen. Sind keine Schriften mit übereinstimmender Höhen- und Seitenabmessung vorhanden, wählt die GDI ein Höhen- und Seitenverhältnis, das mit dem von ihr gewählten übereinstimmt.

DC: Bezeichnet den Gerätekontext, der das Schriftenabbilder-Flag enthält.

Flag: Bestimmt, ob der Schriftenabbilder versucht, das Höhen- und Seitenverhältnis an das Gerät anzugleichen. Wenn das erste Bit auf 1 gesetzt ist, wählt der Abbilder nur Schriften aus, deren x- und y-Abmessungen exakt mit denen des angegebenen Geräts übereinstimmen.

Rückgabewert Es wird der vorherige Wert des Schriftenabbilder-Flags zurückgeliefert.

Hinweis Die übrigen Bits des Parameters *Flag* müssen 0 sein.

SetMenu

Syntax
```
BOOL SetMenu(HWND Wnd, HMENU Menu);
```

Beschreibung Diese Funktion ändert das Menü des angegebenen Fensters in das vom Parameter *Menu* bezeichnete Menü. Ist *Menu* 0, wird das aktuelle Menü des Fensters vom Bildschirm gelöscht. *SetMenu* bewirkt, daß das Fenster neu gezeichnet wird, um die Menüänderung widerzuspiegeln.

Wnd: Bezeichnet das Fenster, dessen Menü geändert werden soll.

Menu: Bezeichnet das neue Menü.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Menü geändert wurde, andernfalls ist er Null.

Hinweis *SetMenu* entfernt ein vorheriges Menü nicht aus dem Speicher. Um dies zu bewirken, sollte eine Anwendung die Funktion *DestroyMenu* aufrufen.

Querverweis *DestroyMenu*

SetMenuItemBitmaps

Syntax
```
BOOL SetMenuItemBitmaps(HMENU Menu, WORD Position,
WORD Flags, HBITMAP BitmapUnchecked,
HBITMAP BitmapChecked);
```

Beschreibung Diese Funktion ordnet die angegebenen Bitmaps einer Menüoption zu. Je nachdem, ob die Menüoption markiert ist

oder nicht, stellt Windows das entsprechende Bitmap neben der Menüoption dar.

Menu: Bezeichnet das zu bearbeitende Menü.

Position: Bestimmt die Menüoption, die verändert werden soll. Wenn *Flags* auf *MF_BYPOSITION* gesetzt ist, bestimmt *Position* die Position der Menüoption, die erste Option im Menü steht an Position 0. Wenn *Flags* auf *MF_BYCOMMAND* gesetzt ist, bestimmt *Position* die Befehls-ID der Menüoption.

Flags: Bestimmt die Interpretation des Parameters *Position*. Er kann auf *MF_BYCOMMAND* (Voreinstellung) oder *MF_BYPOSITION* gesetzt werden.

BitmapUnchecked: Bezeichnet das Bitmap, das dargestellt werden soll, wenn die Menüoption nicht markiert ist.

BitmapChecked: Bezeichnet das Bitmap, das dargestellt werden soll, wenn die Menüoption markiert ist.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Ist einer der Parameter *BitmapUnchecked* oder *BitmapChecked* 0, stellt Windows für das zugehörige Attribut nichts neben der Menüoption dar. Sind beide Parameter 0, verwendet Windows die voreingestellte Auswahlmarkierung, wenn die Option markiert wird und löscht die Auswahlmarkierung, wenn die Option nicht markiert ist. Wenn das Menü entfernt und freigegeben wird, werden diese Bitmaps nicht freigegeben, hierfür ist die Anwendung verantwortlich.

Von der Funktion *GetMenuCheckMarkDimensions* werden die Dimensionen der für Menüoptionen verwendeten Standardauswahlmarkierung geliefert. Die Anwendung sollte diese Werte benutzen, um die geeignete Größe für Bitmaps, die mit dieser Funktion gesetzt werden, zu bestimmen.

SetMessageQueue

Syntax
```
BOOL SetMessageQueue(int Msg);
```

Beschreibung Diese Funktion erzeugt eine neue Botschaftswarteschlange. Sie ist nützlich in Anwendungen, die eine Warteschlange mit mehr als acht Botschaften benötigen (Maximalgröße der Standardwarteschlange). Der Parameter *Msg* bestimmt die Größe der neuen Warteschlange. Die Funktion muß von der Funktion *WinMain* einer Anwendung aufgerufen werden, bevor Fenster erzeugt und Botschaften gesendet werden. Die Funktion *SetMessageQueue* entfernt die alte Warteschlange zusammen mit allen Botschaften, die diese eventuell enthält.

Msg: Bestimmt die maximale Anzahl von Botschaften, die die neue Warteschlange enthalten kann.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion eine neue Warteschlange erstellt hat, andernfalls ist er Null.

Hinweis Ist der Rückgabewert 0, hat die Anwendung keine Warteschlange mehr, da die Funktion *SetMessageQueue* die ursprüngliche Warteschlange löscht, bevor sie versucht, eine neue zu erzeugen. Die Anwendung muß *SetMessageQueue* solange mit einer kleineren Warteschlangen-Größe aufrufen, bis die Funktion einen Wert ungleich 0 zurückgibt.

SetMetaFileBits

Syntax
```
HANDLE SetMetaFileBits(HANDLE Mem);
```

Beschreibung Diese Funktion erstellt ein Speicher-Metafile aus den Daten in dem durch den Parameter *Mem* bezeichneten Speicherblock.

Mem: Bezeichnet den globalen Speicherblock, der die Metafile-Daten enthält. Die Daten müssen mit der Funktion *GetMetaFileBits* erzeugt worden sein.

Rückgabewert Bei erfolgreicher Ausführung der Funktion wird ein Bezeichner des Speicher-Metafile zurückgeliefert, andernfalls ist der Rückgabewert 0.

Hinweis Nachdem die Funktion *SetMetaFileBits* zurückgekehrt ist, sollte für Bezüge auf das Metafile das zurückgelieferte anstelle des durch den Parameter *Mem* bezeichneten Handles verwendet werden.

SetPaletteEntries

Syntax
```
WORD SetPaletteEntries(HPALETTE Palette,
WORD StartIndex, WORD NumEntries,
LPPALETTEENTRY PaletteEntries);
```

Beschreibung Diese Funktion setzt RGB-Farbwerte und Flags in einer Reihe von Einträgen in einer virtuellen Palette.

Palette: Bezeichnet die virtuelle Palette.

StartIndex: Bestimmt den ersten Eintrag in der virtuellen Palette, der gesetzt werden soll.

NumEntries: Bestimmt die Anzahl der Einträge in der virtuellen Palette, die gesetzt werden sollen.

PaletteEntries: Zeigt auf ein Array von *PALETTEENTRY*-Datenstrukturen, das RGB-Werte und Flags enthält.

Rückgabewert Bei erfolgreicher Ausführung wird die Zahl der Einträge, die in der virtuellen Palette gesetzt wurden, zurückgeliefert, andernfalls 0.

Hinweis Wenn die virtuelle Palette in einen Gerätekontext ausgewählt ist, während die Anwendung *SetPaletteEntries* aufruft, haben die Veränderungen solange keine Auswirkung, bis die Anwendung *RealizePalette* aufruft.

SetParent

Syntax
```
HWND SetParent(HWND WndChild, HWND WndNewParent);
```

Beschreibung Diese Funktion ändert das übergeordnete Fenster eines untergeordneten Fensters. Ist das vom Parameter *WndChild* bezeichnete Fenster sichtbar, führt Windows eine entsprechende Neudarstellung durch.

WndChild: Bezeichnet das untergeordnete Fenster.

WndNewParent: Bezeichnet das übergeordnete Fenster.

Rückgabewert Es wird das vorherige übergeordnete Fenster zurückgeliefert.

SetPixel

Syntax
```
DWORD SetPixel(HDC DC, int X, int Y, COLORREF Color);
```

Beschreibung Diese Funktion setzt ein Pixel an dem durch die Parameter *X* und *Y* bestimmten Punkt mit der nächstmöglichen Annäherung an den im Parameter *Color* vorgegebenen Farbwert.

Der Punkt muß innerhalb der Clipping-Region liegen. Ist dies nicht der Fall, wird die Funktion ignoriert.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate des zu setzenden Punktes.

Y: Bestimmt die virtuelle y-Koordinate des zu setzenden Punktes.

Color: COLORREF mit der Darstellungsfarbe für den Punkt.

Rückgabewert Der Rückgabewert bestimmt den RGB-Farbwert der Farbe, die zur Darstellung des Punktes tatsächlich verwendet wurde. Der Wert kann sich von dem durch den Parameter *Color* vorgegebenen unterscheiden, wenn ein Annäherungswert für diese Farbe benutzt wird. Wenn die Funktion fehlschlägt (z.B. weil der Punkt außerhalb der Clipping-Region liegt), ist der Rückgabewert -1.

Hinweis Nicht alle Geräte unterstützen diese Funktion (siehe die Beschreibung der Funktion *GetDeviceCaps*).

SetPolyFillMode

Syntax
```
int SetPolyFillMode(HDC DC, int PolyFillMode);
```

Beschreibung Diese Funktion setzt den Polygonfüllmodus für GDI-Funktionen, die den Polygon-Algorithmus einsetzen, um innenliegende Punkte zu berechnen.

DC: Bezeichnet den Gerätekontext.

PolyFillMode: Bestimmt den neuen Füllmodus. Der Parameter kann die Werte *ALTERNATE* und *WINDING* annehmen. Einzelheiten hierzu finden sich im folgenden »Hinweis«.

Rückgabewert Bei erfolgreicher Ausführung wird der vorherige Füllmodus zurückgegeben, andernfalls ist der Rückgabewert 0.

Hinweis Im allgemeinen unterscheiden sich die Modi nur in den Fällen, in denen ein komplexes, überlappendes Polygon ausgefüllt werden muß (zum Beispiel ein fünfseitiges Polygon, das einen fünfzackigen Stern mit einem Fünfeck in der Mitte bildet). In solchen Fällen füllt der Modus *ALTERNATE* alternierend die einzelnen abgeschlossenen Bereiche innerhalb des Polygons (also die Zacken des Sterns), der Modus *WINDING* dagegen füllt den gesamten Bereich aus (hier: die Sternzacken und das Fünfeck).

Wenn der Füllmodus *ALTERNATE* ist, füllt die GDI den Bereich zwischen geradzahligen und ungeradzahligen Polygon-Seiten auf jeder Scan-Zeile. Das heißt, die GDI füllt den Bereich zwischen der ersten und der zweiten Seite, zwischen der dritten und vierten Seite, und so weiter.

Zum Ausfüllen aller Bereiche bewirkt der Modus *WINDING*, daß die GDI eine Umrandung mit nicht überlappenden Seiten berechnet und zeichnet. Zum Beispiel wird das fünfseitige Polygon, das den Stern bildet, als zehnseitiges Polygon mit nicht überlappenden Seiten gezeichnet; der entstehende Stern wird ausgefüllt.

SetProp

Syntax
```
BOOL SetProp(HWND Wnd, LPSTR Str, HANDLE Data);
```

Beschreibung Diese Funktion fügt einen neuen Eintrag zur Eigenschaftsliste des angegebenen Fensters hinzu oder verändert einen darin vorhandenen Eintrag. Wenn der vom Parameter *Str* bezeichnete String noch nicht in der Liste existiert, fügt *SetProp* einen neuen Eintrag zu der Liste hinzu, der neue Eintrag enthält den

String und das Handle. Andernfalls ersetzt die Funktion das aktuelle Handle des Strings durch *Data*. Der Parameter *Data* kann jeden für die Anwendung nützlichen 16-Bit-Wert enthalten.

Wnd: Bezeichnet das Fenster, dessen Eigenschaftsliste den neuen Eintrag aufnehmen soll.

Str: Zeigt auf einen nullterminierten String oder ein Atom, das einen String bezeichnet. Wenn ein Atom gegeben ist, muß es vorher mit der Funktion *AddAtom* erstellt worden sein. Das Atom, ein 16-Bit-Wert, muß im niederwertigen WORD von *Str* plaziert werden, das höherwertige WORD muß 0 sein.

Data: Bezeichnet ein Daten-Handle, dessen zugehörige Daten in die Eigenschaftsliste kopiert werden soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Daten-Handle und der String zur Eigenschaftsliste hinzugefügt wurden, andernfalls ist er Null.

Hinweis Bevor sie das Fenster entfernt und freigibt (also bevor sie die Botschaft *WM_DESTROY* verarbeitet), ist die Anwendung für das Löschen aller Einträge verantwortlich, die sie zur Eigenschaftsliste hinzugefügt hat.

Die Funktion *RemoveProp* dient zum Löschen von Einträgen aus einer Eigenschaftsliste.

SetRect

Syntax
```
void SetRect(LPRECT Rect, int X1, int Y1, int X2,
int Y2);
```

Beschreibung Diese Funktion erzeugt ein neues Rechteck, indem sie die *RECT*-Datenstruktur, auf die der Parameter *Rect* zeigt, ausfüllt. Die Koordinaten werden durch die Parameter *X1*, *Y1*, *X2* und *Y2* bestimmt.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die die neuen Rechteckkoordinaten aufnehmen soll.

X1, Y1: Bestimmt die x- bzw. y-Koordinate der oberen linken Ecke.

X2, Y2: Bestimmt die x- bzw. y-Koordinate der unteren rechten Ecke.

Rückgabewert Keiner.

Hinweis Die Breite des Rechtecks, die durch den Absolutbetrag von *X2-X1* gegeben ist, darf 32767 Einheiten nicht überschreiten. Dieser Grenzwert gilt auch für die Höhe des Rechtecks (*Y2 - Y1*).

SetRectEmpty

Syntax
```
void SetRectEmpty(LPRECT Rect);
```

Beschreibung Diese Funktion erzeugt ein leeres Rechteck (alle Koordinaten sind 0).

Rect: Zeigt auf eine *RECT*-Datenstruktur, die das leere Rechteck aufnehmen soll.

Rückgabewert Keiner.

SetRectRgn

Syntax
```
void SetRectRgn(HRGN Rgn, int X1, int Y1, int X2,
int Y2);
```

Beschreibung Diese Funktion erzeugt eine rechteckige Region. Im Unterschied zu *CreateRectRgn* ruft sie jedoch nicht die lokale Speicherverwaltung auf, sondern verwendet stattdessen den Speicherplatz, der für die mit dem Parameter *Rgn* bezeichnete Region reserviert ist.

Rgn: Bezeichnet die Region.

X1, Y1: Bestimmt die x- bzw. y-Koordinate der oberen linken Ecke der rechteckigen Region.

X2, Y2: Bestimmt die x- bzw. y-Koordinate der unteren rechten Ecke der rechteckigen Region.

Hinweis Diese Funktion sollte anstelle der Funktion *CreateRectRgn* benutzt werden, um Aufrufe der lokalen Speicherverwaltung zu vermeiden.

Querverweis *CreateRectRgn*

SetResourceHandler

Syntax
```
FARPROC SetResourceHandler(HANDLE Instance,
LPSTR ResType, FARPROC LoadFunc);
```

Beschreibung Diese Funktion bereitet eine Funktion zum Laden von Ressourcen vor. Sie wird intern von Windows verwendet, um berechnete Ressourcen zu implementieren und kann für Anwendungen nützlich sein, die ihre eigenen Ressourcentypen bearbeiten. Die Verwendung dieser Funktion ist nicht unbedingt erforderlich.

Der Parameter *LoadFunc* zeigt auf eine von der Anwendung bereitgestellte Callback-Funktion. Diese erhält Informationen über die Ressource, die gesperrt werden soll, und kann sie wie

gewünscht weiterverarbeiten. Nachdem sie zurückgekehrt ist, versucht *LockResource*, die Ressource noch einmal zu sperren.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei die Ressource enthält.

ResType: Zeigt auf einen Integer, der einen Ressourcentyp bezeichnet.

LoadFunc: Ist die Adresse der Prozedurinstanz der von der Anwendung bereitgestellten Callback-Funktion.

Rückgabewert — Der Rückgabewert zeigt auf die von der Anwendung bereitgestellte Funktion.

Callback-Funktion

```
FARPROC FAR PASCAL LoadFunc (HANDLE Mem,
HANDLE Instance, HANDLE ResInfo);
```

LoadFunc: Platzhalter für den Namen der in der Anwendung definierten Funktion. Der tatsächliche Funktionsname muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden.

Mem: Bezeichnet eine gespeicherte Ressource.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei die Ressource enthält.

ResInfo: Bezeichnet die Ressource. Es wird davon ausgegangen, daß die Ressource vorher mit der Funktion *FindResource* erzeugt wurde.

Hinweis — Der Parameter *Mem* ist NULL, wenn die Ressource noch nicht geladen war. Wenn ein Versuch, einen durch *Mem* bezeichneten Block zu sperren, nicht gelingt, bedeutet das, daß die Ressource verworfen wurde und neu geladen werden muß.

Die Adresse der Dialogfunktion, die als Parameter *LoadFunc* übergeben wurde, muß mit der Funktion *MakeProcInstance* erzeugt werden.

SetROP2

Syntax

```
int SetROP2(HDC DC, int DrawMode);
```

Beschreibung — Diese Funktion setzt den aktuellen Zeichenmodus. Die GDI verwendet den Zeichenmodus, um Stifte und die Innenflächen von ausgefüllten Objekten mit den Farben zu kombinieren, die sich schon auf der Bildschirmoberfläche befinden. Der Modus legt fest, wie die Farbe des Stiftes oder des Flächeninneren und die Farbe, die sich schon auf der Bildschirmfläche befindet, zu einer neuen Farbe kombiniert werden.

DC: Bezeichnet den Gerätekontext.

DrawMode: Bestimmt den neuen Zeichenmodus. Er kann jeden der im Abschnitt »R2_XXXX« in Kapitel 4 »Windows-Stile und -Konstanten« beschriebenen Werte annehmen.

Rückgabewert Es wird der vorherige Zeichenmodus zurückgeliefert.

Hinweis Zeichenmodi legen fest, wie die GDI Quell- und Zielfarben kombiniert, wenn mit dem aktuellen Stift gezeichnet wird. Die Zeichenmodi sind eigentlich binäre Rasteroperationscodes, die alle 16 möglichen booleschen Funktionen zweier Variablen repräsentieren. Zum Zeichen werden die binären UND- (&), ODER- (|) und Exklusives-ODER-(^)Operatoren und die einstellige NOT-Operation verwendet, gegebenfalls in Kombination. Der Zeichenmodus ist nur für Geräte mit Rasterpunkten geeignet. Auf vektororientierten Geräten ist er nicht verfügbar. Weitere Informationen bietet die Beschreibung der Funktion *GetDeviceCaps* sowie der Abschnitt »R2_XXXX« in Kapitel 4 »Windows-Stile und -Konstanten«.

SetScrollPos

Syntax
```
int SetScrollPos(HWND Wnd, int Bar, int Pos,
BOOL Redraw);
```

Beschreibung Diese Funktion setzt die aktuelle Position des Positionszeigers auf die durch den Parameter *Pos* angegebene Position und zeichnet die Bildlaufleiste neu (wenn dies angegeben ist), um die neue Position wiederzugeben.

Wnd: Bezeichnet das Fenster, dessen Bildlaufleiste gesetzt werden soll.

Bar: Bestimmt die zu setzende Bildlaufleiste. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
SB_CTL	Setzt die Position einer selbständigen Bildlaufleiste. In diesem Fall muß der Parameter *Wnd* das Handle einer Bildlaufleiste sein.
SB_HORZ	Setzt die Position der horizontalen Bildlaufleiste eines Fensters.
SB_VERT	Setzt die Position der vertikalen Bildlaufleiste eines Fensters.

Pos: Bestimmt die neue Position. Sie muß innerhalb des Bildlaufbereichs liegen.

Redraw: Bestimmt, ob die Bildlaufleiste neu gezeichnet werden soll, um die neue Position widerzuspiegeln. Ist der Parameter

Redraw ungleich Null, wird die Bildlaufleiste neu gezeichnet. Ist er Null, unterbleibt die Neudarstellung.

Rückgabewert Es wird die vorherige Position des Bildlaufleistensymbols zurückgeliefert.

Hinweis Wenn die Bildlaufleiste von einem nachfolgenden Aufruf einer anderen Funktion neu gezeichnet wird, sollte der Parameter *Redraw* auf *Null* gesetzt werden.

SetScrollRange

Syntax
```
void SetScrollRange(HWND Wnd, int Bar, int MinPos,
int MaxPos, BOOL Redraw);
```

Beschreibung Diese Funktion setzt Minimal- und Maximalpositionswerte für die bezeichnete Bildlaufleiste.

Sie kann außerdem dazu eingesetzt werden, Standardbildlaufleisten anzuzeigen oder zu verbergen. Hierzu müssen die Parameter *MinPos* und *MaxPos* auf 0 gesetzt werden.

Wnd: Bezeichnet ein Fenster oder eine Bildlaufleiste in Abhängigkeit vom Wert des Parameters *Bar*.

Bar: Bestimmt die zu setzende Bildlaufleiste. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
SB_CTL	Setzt den Bereich einer selbständigen Bildlaufleiste. In diesem Fall muß der Parameter *Wnd* das Handle auf eine Bildlaufleiste sein.
SB_HORZ	Setzt den Bereich der horizontalen Bildlaufleiste eines Fensters.
SB_VERT	Setzt den Bereich der vertikalen Bildlaufleiste eines Fensters.

MinPos: Bestimmt die kleinste Bildlaufposition.

MaxPos: Bestimmt die größte Bildlaufposition.

Redraw: Bestimmt, ob die Bildlaufleiste neu gezeichnet werden soll, um die Änderung anzuzeigen. Ist der Parameter *Redraw* ungleich Null, wird die Bildlaufleiste neu gezeichnet, andernfalls nicht.

Hinweis Eine Anwendung sollte diese Funktion nicht zum Ausblenden einer Bildlaufleiste aufrufen, während sie eine Bildlaufleistenbotschaft verarbeitet. Wenn *SetScrollRange* unmittelbar auf die Funktion *SetScrollPos* folgt, sollte der Parameter *Redraw* in *SetScrollPos* auf Null gesetzt werden, um zu verhindern, daß die Bildlaufleiste zweimal gezeichnet wird.

Die Differenz zwischen den durch die Parameter *MinPos* und *MaxPos* festgelegten Werten darf nicht größer als 32767 sein.

SetSoundNoise

Syntax

```
int SetSoundNoise(int Source, int Duration);
```

Beschreibung Diese Funktion setzt Quelle und Dauer eines Klangs am Lautsprecher des Klangausgabegeräts.

Source: Bestimmt die Klangquelle. Er kann einen der folgenden Werte annehmen, wobei *N* der Wert für eine gewünschte Frequenz ist:

Wert	Bedeutung
S_PERIOD512	Quellfrequenz ist N/512 (hoch); weniger heller Klang.
S_PERIOD1024	Quellfrequenz ist N/1024.
S_PERIOD2048	Quellfrequenz ist N/2048 (tief); hellerer Klang.
S_PERIODVOICE	Quellfrequenz von Voice-Kanal 3.
S_WHITE512	Quellfrequenz ist N/512 (hoch); weniger heller Klang.
S_WHITE1024	Quellfrequenz ist N/1024.
S_WHITE2048	Quellfrequenz ist N/2048 (tief); heller Klang.
S_WHITEVOICE	Quellfrequenz von Voice-Kanal 3.

Duration: Legt die Dauer des Klangs fest (in Clock-Ticks).

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war. Ist die Quelle ungültig, ist der Rückgabewert *S_SERDSR*.

SetStretchBltMode

Syntax

```
int SetStretchBltMode(HDC DC, int StretchMode);
```

Beschreibung Diese Funktion setzt den Streckmodus für die *StretchBlt*-Funktion. Der Streckmodus legt fest, welche Scan-Zeilen und/oder -Spalten *StretchBlt* beim Kontrahieren eines Bitmaps entfernt.

DC: Bezeichnet den Gerätekontext.

StretchMode: Bestimmt den neuen Streckmodus. Er kann einen der in folgender Tabelle beschriebenen Werte annehmen:

Ein typischer Verwendungszweck der Modi *BLACKONWHITE* und *WHITEONBLACK* ist es, Vordergrundpixel in Monochrom-Bitmaps zu erhalten. Eine typische Verwendung des Modus *COLORONCOLOR* ist es, die Farben in Farb-Bitmaps zu erhalten.

Wert	Bedeutung
BLACKONWHITE	**and** in den getilgten Zeilen. Dieser Modus erhält schwarze Pixel auf Kosten der weißen durch Verwendung des UND-Operators (&) mit den zu tilgenden und den verbleibenden Zeilen.
COLORONCOLOR	Dieser Modus löscht alle getilgten Zeilen und versucht dabei nicht, ihre Information zu erhalten.
WHITEONBLACK	**or** in den getilgten Zeilen. Dieser Modus erhält weiße Pixel auf Kosten der schwarzen durch den Gebrauch des ODER-Operators (|) mit den zu tilgenden und den verbleibenden Zeilen.

Rückgabewert Der Rückgabewert gibt den vorherigen Streckmodus an. Er kann *BLACKONWHITE*, *COLORONCOLOR* oder *WHITEONBLACK* sein.

SetSwapAreaSize

Syntax
```
LONG SetSwapAreaSize(WORD Size);
```

Beschreibung Diese Funktion erhöht den Speicherumfang, den eine Anwendung für ihre Code-Segmente verwendet. Der maximal verfügbare Speicherumfang ist die Hälfte des freien Speichers, der nach dem Laden von Windows verbleibt.

Size: Bestimmt die Anzahl der 16-Byte-Paragraphen, die eine Anwendung zur Verwendung als Code-Segment anfordert.

Rückgabewert Das niederwertige WORD des Rückgabewertes bestimmt die Anzahl der erhaltenen Paragraphen, die als Speicher für das Code-Segment belegt werden (oder die gegenwärtige Größe, wenn *Size* 0 ist); das höherwertige WORD bestimmt die maximal verfügbare Größe.

Hinweis Legt *Size* eine Größe fest, die die verfügbare überschreitet, setzt diese Funktion die Größe auf den maximal verfügbaren Wert.

Sobald Speicher für den Gebrauch als Code-Segment zugeordnet wurde, kann ihn eine Anwendung nicht mehr durch den Aufruf der Funktion *GlobalAlloc* als Datensegment nutzen.

Der Aufruf von *SetSwapAreaSize* verbessert die Leistungsfähigkeit einer Anwendung, indem sie verhindert, daß Speicherbereiche überschrieben werden. Er vermindert jedoch den verfügbaren Speicher für Datenobjekte und kann die Leistungsfähigkeit anderer Anwendungen einschränken.

Vor dem Aufruf von *SetSwapAreaSize* sollte eine Anwendung *GetNumTasks* aufrufen, um festzustellen, wieviele andere Tasks gerade laufen.

SetSysColors

Syntax

```
void SetSysColors(int Changes, LPINT SysColor,
LONG FAR * ColorValues);
```

Beschreibung

Diese Funktion setzt die Systemfarben für ein oder mehrere Bildschirmelemente. Bildschirmelemente sind die verschiedenen Teile eines Fensters und der Windows-Anzeige, die auf dem Bildschirm des Systems erscheinen. Die Funktion *SetSysColors* ändert die Anzahl der durch den Parameter *Changes* festgelegten Elemente. Sie verwendet dazu den Farben- und Systemfarbenindex, der in den Arrays der Parameter *SysColor* und *ColorValues* enthalten ist.

SetSysColors sendet eine *WM_SYSCOLORCHANGE*-Botschaft an alle Fenster, um sie über den Farbwechsel zu informieren. Außerdem weist sie Windows an, die betroffenen Teile aller momentan sichtbaren Fenster neu zu zeichnen.

Changes: Bestimmt die Anzahl der Systemfarben, die geändert werden sollen.

SysColor: Zeigt auf ein Array von Integerindizes, die die zu ändernden Elemente bestimmen. Die Indexwerte, die verwendet werden können, sind im »Hinweis« beschrieben.

ColorValues: Zeigt auf ein Array vorzeichenloser Integer-Zahlen, das die neuen RGB-Farbwerte für jedes Element enthält.

Hinweis

SetSysColors ändert nur die interne Systemliste. Sie ändert nicht den Abschnitt [COLORS] der Initialisierungsdatei von Windows, WIN.INI. Änderungen gelten nur für die momentane Windows-Sitzung.

Systemfarben sind eine gemeinsam genutzte Ressource. Eine Anwendung sollte eine Farbe nicht ändern, wenn diese nicht in allen Fenstern aller momentan laufenden Anwendungen geändert werden soll. Systemfarben werden für Monochrombildschirme gewöhnlich als verschiedene Graustufen interpretiert. Im folgenden eine Beschreibung der Werte für *SysColor*.

Wert	Bedeutung
COLOR_ACTIVEBORDER	Grenze aktivierter Fenster.
COLOR_ACTIVECAPTION	Überschrift aktivierter Fenster.
COLOR_APPWORKSPACE	Hintergrundfarbe von Multidokumentschnittstellenanwendungen (MDI).
COLOR_BACKGROUND	Desktop.
COLOR_BTNFACE	Flächenschattierung bei Aktionsschaltern.
COLOR_BTNSHADOW	Rahmenschattierung bei Aktionsschaltern.

Wert	Bedeutung
COLOR_BTNTEXT	Text in Aktionsschaltern.
COLOR_CAPTIONTEXT	Text in Titel, Bildgrößenfeld, Bildlaufpfeilfeld.
COLOR_GRAYTEXT	Grauer (nichtaktivierbarer) Text. Diese Farbe wird auf 0 gesetzt, wenn der momentane Bildschirmtreiber einen echten Grauton nicht unterstützt.
COLOR_HIGHLIGHT	Kennzeichnet die gewählte Option in einem Steuerelement.
COLOR_HIGHLIGHTTEXT	Text einer gewählten Option in einem Steuerelement.
COLOR_INACTIVEBORDER	Grenze deaktivierter Fenster.
COLOR_INACTIVECAPTION	Titel deaktivierter Fenster.
COLOR_MENU	Menühintergrund.
COLOR_MENUTEXT	Text in Menüs.
COLOR_SCROLLBAR	Grauer Bereich einer Bildlaufleiste.
COLOR_WINDOW	Fensterhintergrund.
COLOR_WINDOWFRAME	Fensterrahmen.
COLOR_WINDOWTEXT	Text in Fenstern.

SetSysModalWindow

Syntax
```
HWND SetSysModalWindow(HWND Wnd);
```

Beschreibung Diese Funktion macht das gewählte Fenster zu einem systemmodalen Fenster.

Wnd: Bezeichnet das Fenster, das systemmodal werden soll.

Rückgabewert Der Rückgabewert bestimmt das Fenster, das vorher das systemmodale Fenster war.

Hinweis Wenn ein anderes Fenster als das bezeichnete zum aktiven Fenster wird (das systemmodale Fenster öffnet beispielsweise ein Dialogfenster, das zum aktiven Fenster wird), so wird letzteres zum systemmodalen Fenster. Wird das Originalfenster wieder aktiv, ist es systemmodal. Um den systemmodalen Status aufzuheben, muß das systemmodale Fenster geschlossen werden.

SetSystemPaletteUse

Syntax
```
WORD SetSystemPaletteUse(HDC DC, WORD Usage);
```

Beschreibung Diese Funktion erlaubt einer Anwendung, die ganze Systempalette zu verwenden. Die Systempalette enthält per Vorein-

stellung 20 statische Farben, die nicht geändert werden, wenn eine Anwendung ihre virtuelle Palette realisiert.

Der durch den Parameter *DC* festgelegte Gerätekontext muß auf ein Gerät verweisen, das Farbpaletten unterstützt.

DC: Bezeichnet den Gerätekontext.

Usage: Bestimmt den neuen Gebrauch der Systempalette. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
SYSPAL_NOSTATIC	Die Systempalette enthält keine statischen Farben außer Schwarz und Weiß.
SYSPAL_STATIC	Die Systempalette enthält statische Farben, die sich nicht ändern, wenn eine Anwendung ihre virtuelle Palette realisiert.

Rückgabewert

Der Rückgabewert gibt den vorhergehenden Gebrauch der Systempalette an, er ist entweder *SYSPAL_NOSTATIC* oder *SYSPAL_STATIC*.

Hinweis

Eine Anwendung darf diese Funktion nur aufrufen, wenn ihr Fenster den Fokus besitzt.

Ruft eine Anwendung *SetSystemPaletteUse* auf, wobei *Usage* auf *SYSPAL_NOSTATIC* gesetzt ist, behält Windows zwei Einträge in der Systempalette für reines Weiß bzw. reines Schwarz bei.

Wenn *Usage* auf *SYSPAL_NOSTATIC* gesetzt ist, müssen nach dem Aufruf von *SetSystemPaletteUse* folgende Maßnahmen durchgeführt werden:

1. Aufruf von *UnrealizeObject*, damit die GDI die virtuelle Palette komplett neu anlegt.

2. Realisieren der virtuellen Palette.

3. Aufruf von *GetSysColor* , um die gegenwärtigen Systemfarbeinstellungen zu sichern.

4. Aufruf von *SetSysColors*, um die Systemfarben auf sinnvolle Werte zu setzen. Benachbarte oder überlappende Elemente (wie Fensterrahmen und -grenzen) sollten beispielsweise auf schwarz oder weiß gesetzt werden.

5. Ausgabe der Rundrufbotschaft *WM_SYSCOLORCHANGE*, um anderen Fenstern die Verwendung der neuen Systemfarben zu ermöglichen.

Wenn das Fenster der Anwendung den Fokus verliert oder schließt, muß die Anwendung folgende Schritte ausführen:

1. Aufruf von *SetSystemPaletteUse*, wobei der Parameter *Usage* auf *SYSPAL_STATIC* gesetzt wird.

2. Aufruf von *UnrealizeObject*, damit die GDI die virtuelle Palette bei ihrem Aufruf komplett neu anlegt.

3. Realisieren der virtuellen Palette.

4. Setzen der Systemfarben auf ihre ursprünglichen Werte.

5. Ausgabe der Rundrufbotschaft *WM_SYSCOLORCHANGE*.

Querverweis *GetSysColor, SetSysColors, UnrealizeObject, WM_SYSCOLORCHANGE*

SetTextAlign

Syntax
```
WORD SetTextAlign(HDC DC, WORD Flags);
```

Beschreibung Diese Funktion setzt die Textausrichtungsflags für den gegebenen Gerätekontext. Die Funktionen *TextOut* und *ExtTextOut* verwenden diese Flags beim Positionieren eines Textstrings auf dem Bildschirm oder auf einem Gerät. Die Flags legen das Verhältnis zwischen einem speziellen Punkt und einem Rechteck, das den Text umschließt, fest. Die Koordinaten dieses Punktes werden als Parameter an die Funktion *TextOut* übergeben. Das Rechteck, das den Text umschließt, wird durch die aufeinanderfolgenden Zeichenzellen im Textstring gebildet.

DC: Bezeichnet den Gerätekontext.

Flags: Bestimmt eine Maske mit Werten der folgenden Liste. Für horizontale und vertikale Ausrichtung darf nur jeweils ein Flag gewählt werden. Außerdem kann nur eines der beiden Flags, die die aktuelle Position beeinflussen, gewählt werden:

Wert	Bedeutung
TA_BASELINE	Bestimmt die Ausrichtung des Punktes an der Grundlinie der gewählten Schrift.
TA_BOTTOM	Bestimmt die Ausrichtung des Punktes an der Unterseite des umschließenden Rechtecks.
TA_CENTER	Bestimmt die Ausrichtung des Punktes am horizontalen Mittelpunkt des umschließenden Rechtecks.
TA_LEFT	Bestimmt die Ausrichtung des Punktes an der linken Seite des umschließenden Rechtecks.
TA_NOUPDATECP	Bestimmt, daß die aktuelle Position nicht nach jedem Aufruf der Funktionen *TextOut* oder *ExtTextOut* aktualisiert wird.
TA_RIGHT	Bestimmt die Ausrichtung des Punktes an der rechten Seite des umschließenden Rechtecks.
TA_TOP	Bestimmt die Ausrichtung des Punktes an der Oberseite des umschließenden Rechtecks.

Wert	Bedeutung
TA_UPDATECP	Bestimmt, daß die aktuelle Position nach jedem Aufruf der Funktionen *TextOut* oder *ExtTextOut* aktualisiert wird.

Die Standardwerte sind *TA_LEFT*, *TA_TOP*, und *TA_NOUPDATECP*.

Rückgabewert Der Rückgabewert gibt die vorherige Einstellung für die Textausrichtung an, das niederwertige WORD enthält die horizontale, das höherwertige die vertikale Ausrichtung.

SetTextCharacterExtra

Syntax
```
int SetTextCharacterExtra(HDC DC, int CharExtra);
```

Beschreibung Diese Funktion setzt den Wert für den Zeichenabstand. Die GDI fügt diesen Abstand an jedes Zeichen an – einschließlich der Unterbrechungszeichen – wenn im Gerätekontext eine Textzeile geschrieben wird.

DC: Bezeichnet den Gerätekontext.

CharExtra: Bestimmt die Größe des zusätzlichen Abstandes in virtuellen Einheiten, der an jedes Zeichen angefügt werden soll. Ist der aktuelle Abbildungsmodus nicht *MM_TEXT*, so wird der Parameter *CharExtra* übertragen und auf das nächste Pixel gerundet.

Rückgabewert Der Rückgabewert bestimmt den Wert des vorherigen Zeichenabstands.

SetTextColor

Syntax
```
DWORD SetTextColor(HDC DC, COLORREF Color);
```

Beschreibung Diese Funktion setzt die Textfarbe auf die Farbe, die durch den Parameter *Color* bestimmt ist oder auf die nächstliegende physikalische Farbe, wenn das Gerät die durch *Color* festgelegte Farbe nicht darstellen kann. Die GDI verwendet die Textfarbe, um das Erscheinungsbild jedes Zeichens zu bestimmen, das durch die Funktionen *TextOut* und *ExtTextOut* geschrieben wird. Die GDI verwendet die Textfarbe auch bei der Konvertierung von Bitmaps von Farbe in Monochrom und umgekehrt.

Die Hintergrundfarbe für ein Zeichen wird durch die Funktionen *SetBkColor* und *SetBkMode* bestimmt. In der Beschreibung der Funktionen *BitBlt* und *StretchBlt* finden sich Informationen zur Konvertierung von Farb-Bitmaps.

DC: Bezeichnet den Gerätekontext.

Color: *COLORREF*, der die Textfarbe bestimmt.

Rückgabewert Der Rückgabewert gibt den RGB-Farbwert der vorherigen Textfarbe an.

Querverweis *SetBkColor, SetBkMode*

SetTextJustification

Syntax
```
int SetTextJustification(HDC DC, int BreakExtra,
int BreakCount);
```

Beschreibung Diese Funktion bereitet die GDI darauf vor, eine Textzeile zu justieren und dabei die durch die Parameter *BreakExtra* und *BreakCount* festgelegten Justierungsparameter zu verwenden. Um einen Text zu justieren, vergibt die GDI zusätzliche Pixel für die Unterbrechungszeichen in einer Textzeile, die von der Funktion *TextOut* geschrieben wird. Das Unterbrechungszeichen, das zur Abgrenzung von Worten verwendet wird, ist üblicherweise das Leerzeichen (ASCII 32), eine Schrift kann jedoch auch ein anderes Zeichen definieren. Mit der Funktion *GetTextMetrics* kann das Unterbrechungszeichen einer Schrift ermittelt werden.

SetTextJustification bereitet die Justierung vor, indem der Wert des anzufügenden Abstandes festgelegt wird. Der Parameter *BreakExtra* bestimmt den Gesamtbetrag des Abstandes, der an die Zeile angefügt wird, in virtuellen Einheiten. Der Parameter *BreakCount* bestimmt, wieviele Unterbrechungszeichen sich in der Zeile befinden. Die nachfolgende Funktion *TextOut* verteilt den zusätzlichen Abstand gleichmäßig auf jedes Unterbrechungszeichen der Zeile.

Die Funktion *GetTextExtent* wird immer zusammen mit der Funktion *SetTextJustification* verwendet. *GetTextExtent* berechnet die Breite einer gegebenen Zeile vor der Justierung. Diese Breite muß bekannt sein, damit ein geeigneter Wert für *BreakExtra* berechnet werden kann.

SetTextJustification kann dazu verwendet werden, eine Zeile zu justieren, die mehrere Folgen in verschiedenen Schriften enthält. In diesem Fall muß die Zeile stückweise durch separate Justierung und Ausgabe jeder Folge erzeugt werden. Da während der Justierung Rundungsfehler auftreten können, verwaltet die GDI einen Rundungsfehlerwert, der den aktuellen Fehler bestimmt. Bei der Justierung einer Zeile, die mehrere Folgen enthält, verwendet *GetTextExtent* automatisch diesen Fehlerwert beim Berechnen des Umfangs der nächsten Folge. Dadurch wird es *TextOut* ermöglicht, den Fehler für die neue Folge zu berücksichtigen. Nachdem eine Zeile vollständig justiert worden ist, muß dieser Fehlerausdruck gelöscht werden,

um zu verhindern, daß er in die nächste Zeile übertragen wird. Dies geschieht dadurch, daß *SetTextJustification* mit auf 0 gesetztem *BreakExtra* aufgerufen wird.

DC: Bezeichnet den Gerätekontext.

BreakExtra: Bestimmt den Gesamtbetrag des Abstandes in virtuellen Einheiten, um den die Textzeile ergänzt wird. Ist der aktuelle Abbildungsmodus nicht *MM_TEXT,* wird der durch den Parameter *BreakExtra* festgelegte Wert übertragen und auf das nächste Pixel gerundet.

BreakCount: Bestimmt die Anzahl der Unterbrechungszeichen in der Zeile.

Rückgabewert Der Rückgabewert ist 1, wenn die Funktion erfolgreich war, ansonsten ist er 0.

Querverweis *GetTextExtent, GetTextMetrics, TextOut*

SetTimer

Syntax
```
WORD SetTimer(HWND Wnd, int IDEvent, WORD Elapse,
FARPROC TimerFunc);
```

Beschreibung Diese Funktion erzeugt ein System-Timer-Ereignis. Tritt ein Timer-Ereignis ein, gibt Windows eine *WM_TIMER*-Botschaft an die durch den Parameter *TimerFunc* bezeichnete anwendungsunterstützte Funktion aus, die dann das Ereignis bearbeiten kann. Hat *TimerFunc* den Wert NULL, werden *WM_TIMER*-Botschaften in der Anwendungswarteschlange plaziert.

Wnd: Bestimmt das Fenster, das dem Timer zugeordnet werden soll. Ist *Wnd* 0, wird dem Timer kein Fenster zugeordnet.

IDEvent: Bestimmt einen von 0 verschiedenen Bezeichner für ein Timer-Ereignis, wenn der Parameter *Wnd* nicht 0 ist. Andernfalls wird dieser Parameter ignoriert.

Elapse: Bestimmt die abgelaufene Zeit zwischen Timer-Ereignissen in Millisekunden.

TimerFunc: Ist die Adresse der Prozedurinstanz der Funktion, die benachrichtigt werden soll, wenn das Timer-Ereignis stattfindet. Wenn *TimerFunc* NULL ist, wird die *WM_TIMER*-Botschaft in der Anwendungswarteschlange plaziert, und das *hwnd*-Element der *MSG*-Struktur enthält den Parameter *Wnd*, der durch den Aufruf von *SetTimer* gegeben ist. Einzelheiten hierzu finden sich im folgenden »Hinweis«.

Rückgabewert Der Rückgabewert bestimmt den Integerbezeichner für das neue Timer-Ereignis (*IDEvent*). Ist der Parameter *Wnd* 0, gibt eine Anwendung diesen Wert an die Funktion *KillTimer* weiter,

um das Timer-Ereignis zu beseitigen. Der Rückgabewert ist 0, wenn der Timer nicht eingerichtet wurde.

Hinweis

Timer sind eine eingeschränkt verfügbare, globale Ressource. Aus diesem Grund ist es wichtig, daß eine Anwendung den durch die Funktion *SetTimer* zurückgegeben Wert überprüft, um sicherzustellen, daß ein Timer verfügbar ist.

Um eine Timer-Funktion zu installieren, muß *SetTimer* eine Adresse der Prozedurinstanz einer Callback-Funktion erhalten, und die Funktion muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der Anwendung exportiert werden. Die Adresse sollte mit Hilfe der Funktion *MakeProcInstance* erzeugt werden.

Callback-Funktion

```
WORD FAR PASCAL TimerFunc(HWND Wnd, WORD Msg,
int IDEvent, DWORD Time);
```

TimerFunc: Platzhalter für den Namen der in der Anwendung definierten Funktion.

Wnd: Bezeichnet das Fenster, das dem Timer-Ereignis zugeordnet wird.

Msg: Bestimmt die *WM_TIMER*-Botschaft.

IDEvent: Bestimmt die ID des Timers.

Time: Bestimmt die aktuelle Systemzeit.

Querverweis

WM_TIMER

SetViewportExt

Syntax

```
DWORD SetViewportExt(HDC DC, int X, int Y);
```

Beschreibung

Diese Funktion legt die x- und y-Abmessungen des Grafikfensters des festgelegten Gerätekontextes fest. Das Grafikfenster definiert zusammen mit dem Gerätekontextfenster, wie die GDI Punkte im virtuellen Koordinatensystem auf Punkte im Koordinatensystem des aktuellen Gerätes abbildet.

Die x- und y-Abmessungen des Grafikfensters bestimmen, in welchem Maß die GDI Einheiten im virtuellen Koordinatensystem strecken oder stauchen muß, um sie an die Einheiten im Gerätekoordinatensystem anzupassen.

Ist beispielsweise die x-Abmessung des Fensters 2 und die x-Abmessung des Grafikfensters 4, so überträgt die GDI zwei virtuelle Einheiten (gemessen an der x-Achse) in vier Geräteeinheiten. Ist die y-Abmessung des Fensters 2 und die y-Abmessung des Grafikfensters -1, so überträgt die GDI zwei virtuelle Einheiten (gemessen an der y-Achse) in eine Geräteeinheit. Die Größen definieren auch die relative Orientierung der x- und y-Achsen in beiden Koordinatensystemen. Sind die

Vorzeichen der zusammengehörenden Fenster- und Grafikfensterabmessungen gleich, haben die Achsen die gleiche Orientierung, sind die Vorzeichen verschieden, so wird die Orientierung umgedreht.

Ist zum Beispiel die y-Abmessung des Fensters 2 und die y-Abmessung des Grafikfensters -1, so stellt die GDI die positive y-Achse im virtuellen Koordinatensystem auf der negativen y-Achse im Gerätekoordinatensystem dar. Sind die x-Abmessungen 2 und 4, stellt die GDI die positive x-Achse im virtuellen Koordinatensystem auf der positiven x-Achse im Gerätekoordinatensystem dar.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die x-Abmessung des Grafikfensters in Geräteeinheiten.

Y: Bestimmt die y-Abmessung des Grafikfensters in Geräteeinheiten.

Rückgabewert Es werden die vorherigen Abmessungen des Grafikfensters zurückgeliefert. Die vorherige y-Abmessung steht im höherwertigen WORD, die vorherige x-Abmessung im niederwertigen. Tritt ein Fehler auf, ist der Rückgabewert 0.

Hinweis Wenn die folgenden Abbildungsmodi gesetzt sind, werden Aufrufe der Funktionen *SetWindowExt* und *SetViewportExt* ignoriert:

MM_HIENGLISH
MM_HIMETRIC
MM_LOENGLISH
MM_LOMETRIC
MM_TEXT
MM_TWIPS

Ist der Modus *MM_ISOTROPIC* gesetzt, muß eine Anwendung vor dem Aufruf von *SetViewportExt* die Funktion *SetWindowExt* aufrufen.

SetViewportOrg

Syntax
```
DWORD SetViewportOrg(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion bestimmt den Ursprung des Grafikfensters für den bezeichneten Gerätekontext. Das Grafikfenster bestimmt zusammen mit dem Gerätekontextfenster, wie die GDI Punkte im virtuellen Koordinatensystem auf Punkte im Koordinatensystem des aktuellen Gerätes abbildet.

Der Ursprung des Grafikfensters markiert den Punkt im Gerätekoordinatensystem, auf den die GDI den Ursprung des

Fensters abbildet. Es ist ein Punkt im virtuellen Koordinatensystem, der durch die Funktion *SetWindowOrg* bestimmt wird.

Die GDI transformiert alle anderen Punkte mit derselben Methode, die erforderlich ist, um den Ursprung des Fensters auf den Ursprung des Grafikfensters zu übertragen.

So werden beispielsweise alle Punkte eines Kreises um den Ursprung des Fensters in einen Kreis um den Ursprung des Grafikfensters herum angeordnet. Ebenso befinden sich alle Punkte einer Linie, die durch den Ursprung des Fensters geht, auf einer Linie durch den Ursprung des Grafikfensters.

DC: Bezeichnet den Gerätekontext.

X, Y: Bestimmt die x- bzw. y-Koordinate des Ursprungs des Grafikfensters in Geräteeinheiten. Der Wert muß innerhalb des Bereichs des Gerätekoordinatensystems liegen.

Rückgabewert Der Rückgabewert ist der vorherige Ursprung des Grafikfensters in Geräteeinheiten. Die y-Koordinate steht im höherwertigen, die x-Koordinate im niederwertigen WORD.

SetVoiceAccent

Syntax
```
int SetVoiceAccent(int Voice, int Tempo, int Volume,
int Mode, int Pitch);
```

Beschreibung Diese Funktion stellt einen festgelegten Ton (Tempo, Lautstärke, Modus und Tonhöhe) in die durch den Parameter *Voice* festgelegte Klang-Warteschlange.

Der neue Ton ersetzt den vorherigen und bleibt solange gültig, bis er ersetzt wird. Ein Ton wird nicht als Note gezählt. Wenn in der Warteschlange kein ausreichender Platz ist, tritt ein Fehler auf; *SetVoiceAccent* läßt in der Warteschlange immer Platz für eine einzelne Synchronisationsmarkierung.

Liegt *Voice* außerhalb des gültigen Bereichs, wird *SetVoiceAccent* ignoriert.

Voice: Bestimmt eine Klangwarteschlange. Die erste Warteschlange wird mit 1 numeriert.

Tempo: Bestimmt die Anzahl von Viertelnoten, die pro Minute gespielt werden. Dies kann ein Wert von 32 bis 255 sein, der Standardwert ist 120.

Volume: Bestimmt den Lautstärkepegel. Er kann jeden Wert zwischen 0 (kleinste Lautstärke) und 255 (größte Lautstärke) annehmen.

Mode: Bestimmt, wie die Töne gespielt werden sollen. Er kann einen der in der folgenden Tabelle beschriebenen Werte annehmen:

Wert	Bedeutung
S_LEGATO:	Der Ton wird für die ganze Dauer gehalten und mit dem Beginn des nächsten Tons ausgeblendet.
S_NORMAL:	Der Ton wird für die ganze Dauer gehalten und bricht ab, bevor der nächste Ton beginnt.
S_STACCATO:	Der Ton wird nur für einen Teil der Dauer gehalten und weist einen deutlichen Abstand zwischen ihm und dem nächsten Ton auf.

Pitch: Bestimmt die Tonhöhe der zu spielenden Töne. Dies kann ein Wert zwischen 0 und 83 sein.

Der Tonhöhenwert wird unter Verwendung von Modulo-84-Arithmetik zu jedem Tonwert addiert.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war, ansonsten ist der Rückgabewert einer der folgenden Werte:

Wert	Bedeutung
S_SERDMD	Ungültiger Modus.
S_SERDTP	Ungültiges Tempo.
S_SERDVL	Ungültige Lautstärke.
S_SERQFUL	Warteschlange voll.

SetVoiceEnvelope

Syntax
```
int SetVoiceEnvelope(int Voice, int Shape,
int RepeatCount);
```

Beschreibung Diese Funktion reiht die Hüllkurve (Kurvenform und Anzahl der Wiederholungen) in die durch den Parameter *Voice* festgelegte Klangwarteschlange ein. Die neue Hüllkurve ersetzt die vorherige und bleibt bis zum nächsten Aufruf von *SetVoiceEnvelope* wirksam. Eine Hüllkurve wird nicht als Ton gezählt.

Wenn in der Warteschlange kein ausreichender Platz ist, tritt ein Fehler auf. *SetVoiceEnvelope* läßt in der Warteschlange immer Platz für eine einzelne Synchronisationsmarkierung. Liegt *Voice* außerhalb des gültigen Bereichs, wird *SetVoiceEnvelope* ignoriert.

Voice: Bestimmt die Klangwarteschlange, der die Hüllkurve zugeordnet wird.

Shape: Bestimmt einen Index auf eine OEM-Kurvenformtabelle.

RepeatCount: Bestimmt die Anzahl der Wiederholungen der Kurvenform während der Dauer einer Note.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war, ansonsten ist der Rückgabewert einer der folgenden Werte:

Wert	Bedeutung
S_SERDRC	Ungültige Anzahl der Wiederholungen.
S_SERDSH	Ungültige Kurvenform.
S_SERQFUL	Warteschlange voll.

SetVoiceNote

Syntax

```
int SetVoiceNote(int Voice, int Value, int Length,
int Cdots);
```

Beschreibung Diese Funktion reiht einen durch die Parameter *Value*, *Length* und *Cdots* bestimmten Ton in die durch den Parameter *Voice* festgelegte Klangwarteschlange ein. Ein Fehler tritt auf, wenn in der Warteschlange kein ausreichender Platz ist. Die Funktion läßt in der Warteschlange immer Platz für eine einzelne Synchronisationsmarkierung.

Voice: Bestimmt die Klangwarteschlange, der der Ton zugeordnet wird. Ist *Voice* außerhalb des gültigen Bereichs, wird *SetVoiceNote* ignoriert.

Value: Bestimmt eine von 84 möglichen Noten (sieben Oktaven). Ist *Value* 0, wird eine Pause angenommen.

Length: Bestimmt den Kehrwert der Dauer des Tons. 1 setzt beispielsweise eine ganze Note fest, 2 eine halbe Note, 4 eine viertel Note, usw.

Cdots: Bestimmt die Dauer des Tons in Punkten. Die Dauer ist gleich *Length* * (*Cdots* * 3/2).

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war, ansonsten ist der Rückgabewert einer der folgenden Werte:

Wert	Bedeutung
S_SERDCC	Ungültige Punktanzahl.
S_SERDLN	Ungültige Tonlänge.
S_SERDNT	Ungültige Note.
S_SERQFUL	Warteschlange voll.

SetVoiceQueueSize

Syntax

```
int SetVoiceQueueSize(int Voice, int Bytes);
```

Beschreibung Diese Funktion reserviert die durch den Parameter *Bytes* festgelegte Anzahl von Bytes für die durch den Parameter *Voice* festgelegte Klangwarteschlange.

Ist die Größe der Warteschlange nicht bestimmt, ist der Standardwert 192 Bytes. Dies bietet ungefähr 32 Tönen Platz. Alle Klangwarteschlangen werden im Speicher gesperrt. Die

Warteschlangen können nicht gesetzt werden, während Musik spielt.

Voice: Bestimmt eine Klangwarteschlange.

Bytes: Bestimmt die Anzahl von Bytes für die Klangwarteschlange.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war. Im Fall eines Fehlers sind folgende Werte möglich:

Wert	Bedeutung
S_SERMACT	Musik aktiv.
S_SEROFM	Zu wenig Speicher.

SetVoiceSound

Syntax
```
int SetVoiceSound(int Voice, LONG Frequency,
int Duration);
```

Beschreibung Diese Funktion stellt Frequenz und Dauer in die durch den Parameter *Voice* festgelegte Klangwarteschlange.

Voice: Bestimmt eine Klangwarteschlange. Die erste Klangwarteschlange wird mit 1 bezeichnet.

Frequency: Bestimmt die Frequenz. Das höherwertige WORD enthält die Frequenz in Hertz, das niederwertige WORD enthält die Teilfrequenz.

Duration: Bestimmt die Dauer des Tons in Clock-Ticks.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war. Im Fall eines Fehlers sind folgenden Werte möglich:

Wert	Bedeutung
S_SERDDR	Ungültige Dauer.
S_SERDFQ	Ungültige Frequenz.
S_SERDVL	Ungültige Lautstärke.
S_SERQFUL	Warteschlange voll.

SetVoiceThreshold

Syntax
```
int SetVoiceThreshold(int Voice, int Notes);
```

Beschreibung Diese Funktion setzt den Schwellenwert für die durch *Voice* bestimmte Klangwarteschlange.

Wenn die Anzahl der in der Klangwarteschlange verbleibenden Töne den durch den Parameter *Notes* festgelegten Wert unterschreitet, wird das Schwellenflag gesetzt. Ist beim Aufruf

von *SetVoiceTreshold* die Länge der Warteschlange kleiner als *Notes*, wird das Flag nicht gesetzt.

Um den momentanen Schwellenstatus zu ermitteln, sollte die Funktion *GetThresholdStatus* aufgerufen werden.

Voice: Bestimmt die zu bearbeitende Klangwarteschlange.

Notes: Bestimmt die Anzahl der Töne für den Schwellenwert.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich ist. Er ist 1, wenn die Anzahl der Töne in *Notes* außerhalb des gültigen Bereichs liegt.

Querverweis *GetThresholdStatus*

SetWindowExt

Syntax
```
DWORD SetWindowExt(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion setzt die x- und y-Abmessungen des Fensters, das dem bezeichneten Gerätekontext zugeordnet ist.

Das Fenster legt zusammen mit dem Gerätekontextfenster fest, wie die GDI Punkte im virtuellen Koordinatensystem auf Punkte im Gerätekoordinatensystem abbildet. Die x- und y-Abmessungen des Fensters bestimmen, in welchem Maß die GDI Einheiten im virtuellen Koordinatensystem strecken oder stauchen muß, um die Einheiten dem Gerätekoordinatensystem anzupassen.

Wenn zum Beispiel die x-Abmessung des Fensters 2 und die x-Abmessung des Grafikfensters 4 sind, überträgt die GDI zwei virtuelle Einheiten (gemessen an der x-Achse) in vier Geräteeinheiten. Ist die y-Abmessung des Fensters 2 und die y-Abmessung des Grafikfensters -1, überträgt die GDI zwei virtuelle Einheiten (gemessen an der y-Achse) in eine Geräteeinheit.

Die Größen definieren auch die relative Orientierung der x- und y-Achsen in beiden Koordinatensystemen. Sind die Vorzeichen der zusammengehörenden Fenster- und Grafikfenstergrößen gleich, haben die Achsen die gleiche Orientierung, sind die Vorzeichen verschieden, wird die Orientierung umgedreht. Ist z.B. die y-Abmessung des Fensters 2 und die y-Abmessung des Grafikfensters -1, so bildet die GDI die positive y-Achse im virtuellen Koordinatensystem auf die negative y-Achse im Gerätekoordinatensystem ab. Sind die x-Abmessungen 2 und 4, bildet die GDI die positive x-Achse im virtuellen Koordinatensystem auf die positive x-Achse im Gerätekoordinatensystem ab.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die x-Abmessung des Fensters in virtuellen Einheiten.

Y: Bestimmt die y-Abmessung des Fensters in virtuellen Einheiten.

Rückgabewert Der Rückgabewert gibt die vorherigen Abmessungen des Fensters in virtuellen Einheiten an. Die y-Abmessung steht im höherwertigen, die x-Abmessung im niederwertigen WORD. Tritt ein Fehler auf, ist der Rückgabewert 0.

Hinweis Ist einer der folgenden Abbildungsmodi gesetzt, werden Aufrufe der Funktionen *SetWindowExt* und *SetViewportExt* ignoriert:

MM_HIENGLISH	*MM_LOMETRIC*
MM_HIMETRIC	*MM_TEXT*
MM_LOENGLISH	*MM_TWIPS*

Wenn der Modus *MM_ISOTROPIC* gesetzt ist, muß eine Anwendung die Funktion *SetWindowExt* aufrufen, bevor sie *SetViewportExt* aufruft.

SetWindowLong

Syntax
```
LONG SetWindowLong(HWND Wnd, int Index, LONG NewLong);
```

Beschreibung Diese Funktion ändert ein Attribut des durch den Parameter *Wnd* bezeichneten Fensters.

Wnd: Bezeichnet das Fenster.

Index: Bestimmt den Byte-Offset des zu ändernden Attribut-WORD. Er kann einen der folgenden Werte annnehmen:

Wert	Bedeutung
GWL_EXSTYLE	Setzt einen neuen erweiterten Fensterstil.
GWL_STYLE	Setzt einen neuen Fensterstil.
GWL_WNDPROC	Setzt einen neuen Zeiger auf die Fensterfunktion.

NewLong: Bestimmt den Austauschwert.

Rückgabewert Der Rückgabewert ist der vorherige Wert des bezeichneten LONG-Wertes.

Hinweis Wird *SetWindowLong* mit dem Index *GWL_WNDPROC* verwendet, um eine neue Fensterfunktion zu setzen, muß diese Funktion die Form einer Fensterfunktion haben und in der Moduldefinitionsdatei der Anwendung exportiert werden (siehe die Beschreibung der Funktion *RegisterClass*).

Der Aufruf von *SetWindowLong* mit dem *GCL_WNDPROC*-Index erzeugt eine Unterklasse der zur Erzeugung des Fensters verwendeten Fensterklasse. Eine Anwendung sollte nicht versuchen, eine Fensterunterklasse (Kombinationsfenster oder

Aktionsschalter) für Standard-Windows-Steuerelemente zu erzeugen.

Um auf zusätzliche Vier-Byte-Werte zuzugreifen, die beim Erzeugen der Fensterklasse reserviert wurden, wird ein positiver Byte-Offset ab dem durch den Parameter *Index* festgelegten Index verwendet. Dieser Index ist 0 für den ersten Vier-Byte-Wert im zusätzlichen Speicherbereich, 4 für den nächsten Vier-Byte-Wert, usw.

SetWindowOrg

Syntax
```
DWORD SetWindowOrg(HDC DC, int X, int Y);
```

Beschreibung Diese Funktion setzt den Ursprung des Fensters des angegebenen Gerätekontextes. Das Fenster legt zusammen mit dem Grafikfenster des Gerätekontextes fest, wie die GDI Punkte im virtuellen Koordinatensystem auf Punkte im Gerätekoordinatensystem abbildet.

Der Ursprung des Fensters markiert den Punkt im virtuellen Koordinatensystem, von dem aus die GDI den Ursprung des Grafikfensters abbildet. Dieser ist ein Punkt im Gerätekoordinatensystem, der durch die Funktion *SetWindowOrg* bestimmt wird. Die GDI transformiert alle anderen Punkte mit derselben Methode, die erforderlich ist, um den Ursprung des Fensters auf den Ursprung des Grafikfensters zu übertragen.

So werden beispielsweise alle Punkte eines Kreises um den Ursprung des Fensters in einem Kreis um den Ursprung des Grafikfensters herum angeordnet. Ebenso befinden sich alle Punkte einer Linie, die durch den Ursprung des Fensters verläuft, auf einer Linie durch den Ursprung des Grafikfensters.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate des neuen Fensterursprungs.

Y: Bestimmt die virtuelle y-Koordinate des neuen Fensterursprungs.

Rückgabewert Der Rückgabewert gibt den vorherigen Ursprung des Fensters an. Die y-Koordinate steht im höherwertigen, die x-Koordinate im niederwertigen WORD.

SetWindowPos

Syntax
```
void SetWindowPos(HWND Wnd, HWND WndInsertAfter, int X,
int Y, int cx, int cy, WORD Flags);
```

Beschreibung Diese Funktion ändert Größe, Position und Reihenfolge von untergeordneten Fenstern, Pop-up- und Hauptfenstern. Diese

Fenster werden in der Reihenfolge ihres Erscheinens am Bildschirm angeordnet, das oberste Fenster erhält den höchsten Rang und ist das erste Fenster in der Liste. Diese Reihenfolge wird in einer Fensterliste aufgezeichnet.

Wnd: Bezeichnet das Fenster, das positioniert wird.

WndInsertAfter: Bezeichnet das Fenster in der Liste des Fenster-Managers, das dem positionierten Fenster vorangeht.

X: Die x-Koordinate der oberen linken Ecke des Fensters.

Y: Die y-Koordinate der oberen linken Ecke des Fensters.

cx: Bestimmt die neue Breite des Fensters.

cy: Bestimmt die neue Höhe des Fensters.

Flags: Bestimmt einen von acht möglichen 16-Bit-Werten, die Größenverhältnisse und Positionierung des Fensters betreffen. Er muß einen der folgenden Werte annehmen:

Wert	Bedeutung
SWP_DRAWFRAME	Zeichnet einen Rahmen (definiert in der Klassenbeschreibung des Fensters) um das Fenster.
SWP_HIDEWINDOW	Versteckt das Fenster.
SWP_NOACTIVATE	Aktiviert das Fenster nicht.
SWP_NOMOVE	Behält die aktuelle Position bei (ignoriert die Parameter *x* und *y*).
SWP_NOSIZE	Behält die aktuelle Größe bei (ignoriert die Parameter *cx* und *cy*).
SWP_NOREDRAW	Zeichnet Veränderungen nicht neu.
SWP_NOZORDER	Behält die aktuelle Reihenfolge bei (ignoriert den Parameter *WndInsertAfter*).
SWP_SHOWWINDOW	Zeigt das Fenster.

Rückgabewert Keiner.

Hinweis Ist das *SWP_NOZORDER*-Flag nicht gesetzt, plaziert Windows das durch den Parameter *Wnd* bezeichnete Fenster an der Position, die dem durch den Parameter *WndInsertAfter* festgelegten Fenster folgt. Ist *WndInsertAfter* NULL, so plaziert Windows das durch *Wnd* bezeichnete Fenster am Anfang der Liste. Ist *WndInsertAfter* 1, plaziert Windows das durch *Wnd* bezeichnete Fenster am Ende der Liste.

Sind die Flags *SWP_SHOWWINDOW* oder *SWP_HIDEWINDOW* gesetzt, können Bildlauf und Verschieben nicht gleichzeitig erfolgen. Alle Koordinaten für untergeordnete Fenster gelten relativ zur oberen linken Ecke des Client-Bereichs des Hauptfensters.

SetWindowsHook

Syntax

```
FARPROC SetWindowsHook(int FilterType,
FARPROC FilterFunc);
```

Beschreibung

Diese Funktion installiert eine Filterfunktion in einer Kette. Eine Filterfunktion bearbeitet Ereignisse, bevor diese an die Botschaftsschleife in der Funktion *WinMain* einer Anwendung übergeben werden. Eine Kette ist eine verkettete Liste von Filterfunktionen desselben Typs.

FilterType: Bestimmt die zu installierende Systemfilterverwaltung. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
WH_CALLWNDPROC	Installiert einen Fensterfunktionsfilter.
WH_GETMESSAGE	Installiert einen Botschaftsfilter.
WH_JOURNALPLAYBACK	Installiert einen Wiedergabeprotokollfilter.
WH_JOURNALRECORD	Installiert einen Aufzeichnungsprotokollfilter.
WH_KEYBOARD	Installiert einen Tastaturfilter.
WH_MSGFILTER	Installiert einen Botschaftsfilter.
WH_SYSMSGFILTER	Installiert einen systemweiten Botschaftsfilter.

FilterFunc: Ist die Adresse der Prozedurinstanz der zu installierenden Filterfunktion. Einzelheiten finden sich im »Hinweis«.

Rückgabewert

Der Rückgabewert zeigt auf die Adresse der Prozedurinstanz des vorher installierten Filters. Er ist NULL, wenn kein vorheriger Filter existiert. Die Anwendung oder Bibliothek, die die Funktion *SetWindowsHook* aufruft, sollte den Rückgabewert im Datensegment der Bibliothek sichern. Der vierte Parameter der Funktion *DefHookProc* zeigt auf die Position im Speicher, an der die Bibliothek diesen Rückgabewert ablegt.

Hinweis

Die *WH_CALLWNDPROC*-Filterverwaltung beeinflußt die Rechenleistung des Systems. Sie ist nur zum Testen von Anwendungen vorgesehen.

Die Systemfilterverwaltungen sind gemeinsam genutzte Ressourcen. Die Installation einer Filterverwaltung beeinflußt alle Anwendungen. Die meisten Filterverwaltungsfunktionen müssen in Bibliotheken installiert werden, ausgenommen *WH_MSGFILTER*, die task-spezifisch ist. Systemfilterverwaltungen sollten auf Sonderanwendungen beschränkt werden oder als Entwicklungshilfsmittel während des Testens einer Anwendung eingesetzt werden. Bibliotheken, die die Filterverwaltung nicht mehr benötigen, sollten die Filterfunktion entfernen.

Um eine Filterfunktion zu installieren, muß *SetWindowsHook* die Adresse der Prozedurinstanz der Funktion erhalten und die Funktion muß in einer EXPORTS-Anweisung in der Moduldefinitionsdatei der DLL exportiert werden. Bibliotheken können die Prozeduradresse direkt übergeben. Tasks müssen *MakeProcInstance* verwenden, um die Adresse zu liefern. DLLs müssen *GetProcAddress* verwenden, um die Adresse zu liefern.

Der folgende Abschnitt beschreibt, wie die einzelnen Filterverwaltungsfunktionen unterstützt werden.

WH_CALLWNDPROC Filterfunktion

Windows ruft die *WH_CALLWNDPROC*-Filterfunktion immer dann auf, wenn die Funktion *SendMessage*, nicht aber wenn die Funktion *PostMessage* aufgerufen wird.

```
void FAR PASCAL FilterFunc(int Code, WORD wParam,
DWORD lParam)
```

FilterFunc: Platzhalter für den von der Anwendung oder Bibliothek bereitgestellten Funktionsnamen.

Code: Bestimmt, ob die Filterfunktion die Botschaft bearbeiten oder die Funktion *DefHookProc* aufrufen soll. Ist der Parameter *Code* kleiner als 0, sollte die Filterfunktion die Botschaft ohne weitere Bearbeitung an *DefHookProc* übergeben.

wParam: Bestimmt, ob die Botschaft von der aktuellen Task gesendet wird. Er ist ungleich 0, wenn dies der Fall ist, andernfalls NULL.

lParam: Zeigt auf eine Datenstruktur, die Einzelheiten über die durch den Filter abgefangene Botschaft enthält. Nachstehend werden die Reihenfolge, der Typ und die Beschreibung eines jeden Feldes der Datenstruktur aufgeführt:

Feld	Typ/Beschreibung
hlParam	WORD Enthält das höherwertige WORD des Parameters *lParam* der durch den Filter erhaltenen Botschaft.
llParam	WORD Enthält das niederwertige WORD des Parameters *lParam* der durch den Filter erhaltenen Botschaft.
wParam	WORD Enthält den Parameter *wParam* der durch den Filter erhaltenen Botschaft.
wMsg	WORD Enthält die durch den Filter erhaltene Botschaft.
hWnd	WORD Enthält das Handle des Fensters, das die Botschaft erhalten soll.

Rückgabewert Keiner.

Hinweis Die *WH_CALLWNDPROC*-Filterfunktion kann die Botschaft prüfen oder modifizieren. Wird die Steuerung an Windows zurückgegeben, wird die Botschaft mit allen Modifikationen

der Fensterfunktion übergeben. Die Filterfunktion hat keinen Rückgabewert.

WH_GETMESSAGE-Filterfunktion

Windows ruft die *WH_GETMESSAGE*-Filterfunktion immer dann auf, wenn die Funktion *GetMessage* aufgerufen wird, sofort nachdem *GetMessage* eine Botschaft von einer Anwendungswarteschlange erhalten hat.

```
void FAR PASCAL FilterFunc(int Code, Word wParam,
DWORD lParam)
```

FilterFunc: Platzhalter für den von der Bibliothek bereitgestellten Funktionsnamen.

Code: Bestimmt, ob die Filterfunktion die Botschaft bearbeiten oder die Funktion *DefHookProc* aufrufen soll. Ist der Parameter *Code* kleiner als 0, sollte die Filterfunktion die Botschaft ohne weitere Bearbeitung an *DefHookProc* weitergeben.

wParam: Legt einen NULL-Wert fest.

lParam: Zeigt auf eine Botschaftsstruktur.

Rückgabewert

Keiner.

Hinweise

Die *WH_GETMESSAGE*-Filterfunktion kann die Botschaft prüfen oder modifizieren. Wird die Steuerung dann an Windows zurückgegeben, gibt die Funktion *GetMessage* die Botschaft mit allen Modifikationen an die Anwendung zurück, die sie ursprünglich aufgerufen hat.

WH_JOURNALPLAYBACK-Filterfunktion

Windows ruft die *WH_JOURNALPLAYBACK*-Filterfunktion immer dann auf, wenn eine Anfrage für eine Ereignisbotschaft anliegt. Die Funktion dient zur Bereitstellung einer vorher aufgezeichneten Ereignisbotschaft.

```
DWORD FAR PASCAL FilterFunc(int Code, WORD wParam,
DWORD lParam)
```

FilterFunc: Platzhalter für den von der Bibliothek bereitgestellten Funktionsnamen.

Code: Bestimmt, ob die Filterfunktion die Botschaft bearbeiten oder die Funktion *DefHookProc* aufrufen soll. Ist der Parameter *Code* kleiner als 0, sollte die Filterfunktion die Botschaft ohne weitere Bearbeitung an *DefHookProc* weitergeben.

wParam: Bestimmt einen NULL-Wert.

lParam: Zeigt auf die Botschaft, die durch die Filterfunktion bearbeitet wird.

Hinweis

Die *WH_JOURNALPLAYBACK*-Funktion sollte eine Ereignisbotschaft auf den Parameter *lParam* kopieren. Die Botschaft muß vorher durch den *WH_JOURNALRECORD*-Filter aufge-

zeichnet worden sein, dieser darf die Botschaft nicht modifizieren.

Während die *WH_JOURNALPLAYBACK*-Funktion wirksam ist, ignoriert Windows alle Maus- und Tastatureingaben.

Rückgabewert Der Rückgabewert sollte die Zeitdauer in Clock-Ticks bestimmen, die Windows abwarten soll, bevor die Botschaft bearbeitet wird. Diese Zeit kann berechnet werden, indem die Differenz zwischen den Zeitfeldern in der aktuellen und der vorherigen Ereignisbotschaft gebildet wird. Gibt die Funktion 0 zurück, wird die Botschaft unverzüglich bearbeitet. Wird die Steuerung dann an Windows zurückgegeben, wird die Botschaft weiter bearbeitet.

Wenn der Parameter *Code HC_SKIP* ist, sollte die Filterfunktion die nächste aufgezeichnete Ereignisbotschaft für die Rückgabe bei ihrem nächsten Aufruf vorbereiten.

WH_JOURNALRECORD-Filterfunktion Windows ruft die *WH_JOURNALRECORD*-Filterfunktion immer dann auf, wenn eine Ereignisbotschaft aus der Ereigniswarteschlange bearbeitet wird. Der Filter kann dazu verwendet werden, das Ereignis für eine spätere Wiedergabe aufzuzeichnen.

```
void FAR PASCAL FilterFunc(int Code, WORD wParam,
DWORD lParam)
```

FilterFunc: Platzhalter für den von der Bibliothek bereitgestellten Funktionsnamen.

Code: Bestimmt, ob die Filterfunktion die Botschaft bearbeiten oder die Funktion *DefHookProc* aufrufen soll. Ist der Parameter *Code* kleiner als 0, sollte die Filterfunktion die Botschaft ohne weitere Bearbeitung an *DefHookProc* weitergeben.

wParam: Bestimmt einen NULL-Wert.

lParam: Zeigt auf eine Botschaftsstruktur.

Rückgabewert Keiner.

Hinweise Die *WH_JOURNALRECORD*-Filterfunktion sollte eine Kopie der Ereignisbotschaft für eine spätere Wiedergabe anlegen und die Botschaft nicht modifizieren. Wird die Steuerung dann an Windows zurückgegeben, kann die Botschaft bearbeitet werden.

WH_KEYBOARD-Filterfunktion Windows ruft die *WH_KEYBOARD*-Filterfunktion immer dann auf, wenn eine Anwendung die Funktionen *GetMessage* oder *PeekMessage* aufruft und ein Tastaturereignis (*WM_KEYUP* oder *WM_KEYDOWN*) zu bearbeiten ist.

```
int FAR PASCAL FilterFunc(int Code, WORD wParam,
DWORD lParam)
```

FilterFunc: Platzhalter für den von der Bibliothek bereitgestellten Funktionsnamen.

Code: Bestimmt, ob die Filterfunktion die Botschaft bearbeiten oder die Funktion *DefHookProc* aufrufen soll. Ist dieser Wert *HC_NOREMOVE*, so verwendet die Anwendung die Funktion *PeekMessage* mit der Option *PM_NOREMOVE* und entfernt die Botschaft nicht aus der Systemwarteschlange. Ist dieser Wert kleiner als 0, sollte die Filterfunktion die Botschaft ohne weitere Bearbeitung an *DefHookProc* weitergeben.

wParam: Bestimmt den virtuellen Tastencode der vorgegebenen Taste.

lParam: Legt die Anzahl der Wiederholungen, den Scan-Code, den Code für den Tastenzustand, den vorherigen Tastenstatus und den Kontextcode wie folgt fest:

Bit	Wert
0-15	(niederwertiges WORD) Anzahl der Wiederholungen (wieviele Male ein Tastendruck wiederholt wird, wenn der Anwender die Taste gedrückt hält).
16-23	(niederwertiges Byte des höherwertigen WORD) Scan-Code (OEM-abhängiger Wert).
24	Funktionstaste (1, wenn es eine Funktionstaste ist).
25-26	Nicht verwendet.
27-28	(Kontextcode) 1, wenn die Taste Alt gedrückt war, als die Taste betätigt wurde, 0 in anderen Fällen). Wird von Windows intern verwendet.
30	Vorheriger Tastenstatus (1, wenn die Taste gedrückt war, bevor die Botschaft gesendet wurde, 0, wenn die Taste nicht gedrückt war).
31	Tastenzustand (1, wenn die Taste losgelassen wird, 0, wenn die Taste gedrückt ist).

Bit 0 ist das niederwertige Bit:

Rückgabewert

Der Rückgabewert bestimmt, was mit der Botschaft geschieht. Er ist 0, wenn die Botschaft von Windows bearbeitet werden soll und 1, wenn die Botschaft verworfen werden sollte.

WH_MSG-FILTER-Filterfunktion

Windows ruft die *WH_MSGFILTER*-Filterfunktion immer dann auf, wenn ein Dialogfenster, ein Meldungsfenster oder ein Menü eine Botschaft erhalten hat, bevor diese bearbeitet wird. Der Filter erlaubt einer Anwendung, die Botschaften zu bearbeiten oder zu modifizieren.

Anmerkung: Dies ist der einzige Task-spezifische Filter. Jede Task kann diesen Filter installieren.

```
int FAR PASCAL FilterFunc(int Code, WORD wParam,
DWORD lParam)
```

FilterFunc: Platzhalter für den von der Bibliothek oder Anwendung bereitgestellten Funktionsnamen.

Code: Bestimmt den Typ der bearbeiteten Botschaft. Er muß einen der folgenden Werte annehmen:

Wert	Bedeutung
MSGF_DIALOGBOX	Bearbeitet Botschaften innerhalb einer Dialogfenster- oder Meldungsfensterfunktion.
MSGF_MENU	Bearbeitet Tastatur- und Mausbotschaften in einem Menü. Ist dieser Wert kleiner als 0, sollte die Filterfunktion die Botschaft ohne weitere Bearbeitung an *DefHookProc* weitergeben.

wParam: Bestimmt einen NULL-Wert.

lParam: Zeigt auf die Botschaftsstruktur.

Rückgabewert

Der Rückgabewert ist ungleich 0, wenn die Filterverwaltungsfunktion die Botschaft bearbeitet, andernfalls ist er 0.

WH_SYSMSGFILTER-Filterfunktion

Windows ruft die *WH_SYSMSGFILTER*-Filterfunktion immer dann auf, wenn ein Dialogfenster, ein Meldungsfenster oder ein Menü eine Botschaft erhalten hat und bevor diese Botschaft bearbeitet wird. Der Filter erlaubt einer Anwendung, Botschaften für jede beliebige Anwendung des Systems zu bearbeiten oder zu modifizieren.

```
int FAR PASCAL FilterFunc(int Code, WORD wParam,
DWORD lParam)
```

FilterFunc: Platzhalter für den von der Bibliothek bereitgestellten Funktionsnamen.

Code: Bestimmt den Typ der zu bearbeitenden Botschaft. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
MSGF_DIALOGBOX	Bearbeiten von Botschaften innerhalb der Funktion *DialogBox*.
MSGF_MENU	Bearbeiten von Tastatur- und Mausbotschaften im Menü.
MSGF_MESSAGEBOX	Bearbeiten von Botschaften innerhalb der Funktion *MessageBox*. Ist dieser Wert kleiner als 0, sollte die Filterfunktion die Botschaft ohne weitere Bearbeitung an *DefHookProc* weitergeben.

wParam: Bestimmt einen NULL-Wert.

lParam: Zeigt auf die Botschaftsstruktur.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn die Filterverwaltungsfunktion die Botschaft bearbeitet, andernfalls ist er 0.

Querverweis *DefHookProc*

SetWindowText

Syntax
```
void SetWindowText(HWND Wnd, LPSTR Str);
```

Beschreibung Diese Funktion setzt den String, auf den der Parameter *Str* zeigt, als Überschrift des bezeichneten Fensters.

Ist der Parameter *Wnd* ein Handle auf ein Steuerelement, dann setzt die Funktion *SetWindowText* den Text des Steuerelements.

Wnd: Bezeichnet das Fenster oder das Steuerelement, dessen Text geändert werden soll.

Str: Zeigt auf einen nullterminierten String.

Rückgabewert Keiner.

SetWindowWord

Syntax
```
WORD SetWindowWord(HWND Wnd, int Index, WORD NewWord);
```

Beschreibung Diese Funktion ändert ein Attribut des durch den Parameter *Wnd* bezeichneten Fensters.

Wnd: Bezeichnet das zu modifizierende Fenster.

Index: Bestimmt den Byte-Offset des zu ändernden Attribut-WORD. Er kann auch einen der folgenden Werte annehmen:

Wert	Bedeutung
GWW_HINSTANCE	Instanz-Handle des Moduls, welches das Fenster besitzt.
GWW_ID	Steuerelement des untergeordneten Fensters.

NewWord: Bestimmt den Austauschwert.

Rückgabewert Der Rückgabewert gibt den vorherigen Wert des festgelegten WORD an.

Hinweis Der positive Byte-Offset, der durch den Parameter *Index* festgelegt ist, kann verwendet werden, um auf zusätzliche Zwei-Byte-Werte zuzugreifen, die reserviert wurden, als die Fensterklassenstruktur erzeugt wurde.

Der Byte-Offset ist 0 für den ersten Zwei-Byte-Wert im zusätzlichen Speicher, 2 für den nächsten Zwei-Byte-Wert, usw.

ShowCaret

Syntax `void ShowCaret(HWND Wnd);`

Beschreibung Diese Funktion zeigt das Caret auf dem Bildschirm an der aktuellen Position an. Sobald es sichtbar ist, läßt es Windows in regelmäßigem Zeitabstand aufblinken.

ShowCaret zeigt das Caret nur, wenn es eine aktuelle Form hat und nicht zweimal oder öfter in Folge verborgen wurde. Besitzt das bezeichnete Fenster das Caret nicht, wird es nicht angezeigt. Ist der Parameter *Wnd* 0, zeigt die Funktion *ShowCaret* das Caret nur, wenn es zu einem Fenster der aktuellen Task gehört.

Das Verbergen des Caret geschieht kumulativ. Wurde die Funktion *HideCaret* fünfmal in Folge aufgerufen, muß *ShowCaret* fünfmal aufgerufen werden, um das Caret sichtbar zu machen.

Wnd: Bezeichnet das Fenster, zu dem das Caret gehört oder ist 0, um indirekt das Besitzerfenster in der aktuellen Task zu bezeichnen.

Rückgabewert Keiner.

Hinweis Das Caret ist eine gemeinsam genutzte Ressource. Ein Fenster sollte das Caret nur zeigen, wenn es den Fokus hat oder aktiv ist.

Querverweis *SetCaretBlinkTime*, *SetCaretPos*

ShowCursor

Syntax `int ShowCursor(BOOL Show);`

Beschreibung Diese Funktion zeigt oder verbirgt den Mauszeiger. *ShowCursor* setzt dazu einen internen Anzeigezähler, dessen Wert bestimmt, ob der Mauszeiger gezeigt werden soll. Ist der Parameter *Show* ungleich Null, wird der Anzeigezählerstand um 1 erhöht.

Der Mauszeiger wird nur angezeigt, wenn der Anzeigezählerstand größer als oder gleich 0 ist. Zu Beginn steht der Anzeigezählerstand auf 0, wenn eine Maus installiert ist. Ist dies nicht der Fall, steht er auf -1.

Show: Bestimmt, ob der Anzeigezählerstand erhöht oder verringert wird. Der Anzeigezählerstand wird erhöht, wenn *Show* ungleich Null ist, andernfalls wird er verringert.

Rückgabewert Der Rückgabewert bestimmt den neuen Anzeigezählerstand.

Hinweis Der Mauszeiger ist eine gemeinsam genutzte Ressource. Ein Fenster, das den Mauszeiger verbirgt, sollte den Mauszeiger

anzeigen, bevor der Mauszeiger seinen Client-Bereich verläßt oder bevor es die Steuerung an ein anderes Fenster abgibt.

ShowOwnedPopups

Syntax
```
void ShowOwnedPopups(HWND Wnd, BOOL Show);
```

Beschreibung Diese Funktion zeigt oder verbirgt alle Pop-up-Fenster, die zu dem durch den Parameter *Wnd* bezeichneten Fenster gehören. Ist der Parameter *Show* ungleich Null, werden alle verborgenen Pop-up-Fenster gezeigt. Ist er Null, werden alle sichtbaren Pop-up-Fenster verborgen.

Wnd: Bestimmt das Fenster, zu dem die Pop-up-Fenster gehören, die gezeigt oder verborgen werden sollen.

Show: Bestimmt, ob Pop-up-Fenster versteckt werden oder nicht. Er ist ungleich Null, wenn alle verborgenen Pop-up-Fenster gezeigt werden sollen und Null, wenn alle sichtbaren Pop-up-Fenster verborgen werden sollen.

Rückgabewert Keiner.

ShowScrollBar

Syntax
```
void ShowScrollBar(HWND Wnd, WORD Bar, BOOL Show);
```

Beschreibung Diese Funktion zeigt oder verbirgt eine Bildlaufleiste, abhängig vom Wert des Parameters *Show*. Ist *Show* ungleich Null, wird die Bildlaufleiste angezeigt, ist *Show* Null, wird die Bildlaufleiste verborgen.

Wnd: Bezeichnet ein Fenster, das in seinem Nicht-Client-Bereich eine Bildlaufleiste enthält, wenn der Parameter *Bar SB_HORZ, SB_VERT,* oder *SB_BOTH* ist. Ist *Bar SB_CTL*, bezeichnet *Wnd* eine selbständige Bildlaufleiste.

Bar: Bestimmt, ob die Bildlaufleiste ein Steuerelement oder Teil des Nicht-Client-Bereichs eines Fensters ist. Ist sie Teil des Nicht-Client-Bereichs, zeigt *Bar* auch an, ob die horizontale, die vertikale oder beide Bildlaufleiste(n) betroffen sind. Dieser Parameter muß einen der folgenden Werte annehmen:

Wert	Bedeutung
SB_BOTH	Bestimmt die horizontale und vertikale Bildlaufleiste des Fensters.
SB_CTL	Bestimmt, daß die Bildlaufleiste ein Steuerelement ist.
SB_HORZ	Bestimmt die horizontale Bildlaufleiste des Fensters.
SB_VERT	Bestimmt die vertikale Bildlaufleiste des Fensters.

Show: Bestimmt, ob Windows die Bildlaufleiste verbirgt oder nicht. Ist *Show* Null, wird die Bildlaufleiste verborgen, andernfalls wird die Bildlaufleiste angezeigt.

Rückgabewert Keiner.

Hinweis Eine Anwendung sollte diese Funktion nicht zum Ausblenden einer Bildlaufleiste verwenden, wenn sie gerade eine Bildlaufleistenbotschaft bearbeitet.

ShowWindow

Syntax
```
BOOL ShowWindow(HWND Wnd, int CmdShow);
```

Beschreibung Diese Funktion zeigt oder entfernt das bezeichnete Fenster in Abhängigkeit vom Wert des Parameters *CmdShow*.

Wnd: Bezeichnet das Fenster.

CmdShow: Bestimmt, wie das Fenster gezeigt werden soll. Er muß einen der Werte annehmen, die im »Hinweis« beschrieben sind.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn das Fenster vorher sichtbar war und Null, wenn das Fenster vorher verborgen war.

Hinweis Die Funktion *ShowWindow* darf pro Programm nur einmal mit dem Parameter *CmdShow* aus der Funktion *WinMain* aufgerufen werden. Spätere Aufrufe von *ShowWindow* müssen einen der Werte aus der folgenden Tabelle verwenden. Folgende Werte sind für den Parameter *CmdShow* möglich:

Wert	Bedeutung
SW_HIDE	Verbirgt das Fenster und gibt die Aktivierung an ein anderes Fenster ab.
SW_MINIMIZE	Verkleinert das festgelegte Fenster zum Symbol und aktiviert das Hauptfenster in der Liste des Fenster-Managers.
SW_RESTORE	Wie *SW_SHOWNORMAL*.
SW_SHOW	Aktiviert ein Fenster und zeigt es in seiner aktuellen Größe und Position.
SW_SHOWMAXIMIZED	Aktiviert das Fenster und zeigt es als ein vergrößertes Fenster.
SW_SHOWMINIMIZED	Aktiviert ein Fenster und zeigt es als Symbol.
SW_SHOWMINNOACTIVE	Zeigt das Fenster als Symbol. Das gegenwärtig aktive Fenster bleibt aktiv.
SW_SHOWNA	Zeigt das Fenster in seinem aktuellen Zustand. Das gegenwärtig aktive Fenster bleibt aktiv.

Wert	Bedeutung
SW_SHOWNOACTIVATE	Zeigt ein Fenster in seiner neuesten Größe und Position. Das gegenwärtig aktive Fenster bleibt aktiv.
SW_SHOWNORMAL	Aktiviert ein Fenster und zeigt es an. Ist das Fenster verkleinert oder vergrößert, stellt Windows es in seiner ursprünglichen Größe und Position wieder her.

SizeofResource

Syntax
```
WORD SizeofResource(HANDLE Instance, HANDLE ResInfo);
```

Beschreibung Diese Funktion liefert die Größe der bezeichneten Ressource in Bytes. Sie wird üblicherweise mit der Funktion *AccessResource* verwendet, um Speicher vorzubereiten, der eine Ressource aus einer Datei aufnehmen sollte.

Instance: Bezeichnet die Instanz des Moduls, dessen ausführbare Datei die Ressource enthält.

ResInfo: Bezeichnet die gewünschte Ressource. Von diesem Handle wird angenommen, daß es durch die Funktion *FindResource* erzeugt wurde.

Rückgabewert Der Rückgabewert bestimmt die Anzahl der Bytes in der Ressource. Er ist 0, wenn die Ressource nicht gefunden werden kann.

Hinweis Der zurückgegebene Wert kann wegen der Ausrichtung größer sein als die aktuelle Ressource selbst. Eine Anwendung sollte sich nicht auf diesen Wert als exakte Größe einer Ressource stützen.

Querverweis *AccessResource*

StartSound

Syntax
```
int StartSound(void);
```

Beschreibung Diese Funktion startet die Wiedergabe jeder Klangwarteschlange.

Die Funktion *StartSound* ist nicht destruktiv, sie kann also beliebig oft aufgerufen werden, um die aktuellen Klangwarteschlangen wiederzugeben.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert vom Typ int ist bedeutungslos.

StopSound

Syntax
```
int StopSound(void);
```

Beschreibung Diese Funktion beendet die Wiedergabe aller Klangwarteschlangen und löscht dann die Inhalte der Warteschlangen. Der Sound-Treiber jeder Stimme wird abgeschaltet.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert vom Typ int ist bedeutungslos.

StretchBlt

Syntax
```
BOOL StretchBlt(HDC DestDC, int X, int Y, int Width,
int Height, HDC SrcDC, int XSrc, int YSrc,
int SrcWidth, int SrcHeight, DWORD Rop);
```

Beschreibung Diese Funktion bewegt ein Bitmap von einem Quellrechteck in ein Zielrechteck und streckt oder staucht das Bitmap – falls nötig – um seine Abmessungen dem Zielrechteck anzupassen. *StretchBlt* verwendet den Streckmodus des Zielgerätekontextes, gesetzt durch die Funktion *SetStretchBltMode*, um zu bestimmen, wie das Bitmap gestreckt oder gestaucht werden muß.

StretchBlt bewegt das Bitmap von dem durch den Parameter *SrcDC* bezeichneten Quellgerät zu dem durch den Parameter *DestDC* angegebenen Zielgerät. Die vier Parameter *XSrc, YSrc, SrcWidth* und *SrcHeight* bestimmen den Ursprung und die Abmessungen des Quellrechtecks, die Parameter *X, Y, Width* und *Height* geben den Ursprung und die Abmessungen des Zielrechtecks an. Die durch den Parameter *Rop* festgelegte Rasteroperation bestimmt, wie das Quell-Bitmap und die auf dem Zielgerät vorhandenen Bits kombiniert werden.

StretchBlt erzeugt ein Spiegelbild des Bitmaps, wenn die Vorzeichen der Parameter *SrcWidth* und *Width* oder *SrcHeight* und *Height* voneinander abweichen. Wenn *SrcWidth* und *Width* unterschiedliche Vorzeichen haben, erzeugt die Funktion ein Spiegelbild des Bitmaps entlang der x-Achse. Haben *SrcHeight* und *Height* unterschiedliche Vorzeichen, erzeugt die Funktion ein Spiegelbild des Bitmaps entlang der y-Achse.

DestDC: Bezeichnet den Gerätekontext, der das Bitmap aufnehmen soll.

X: Bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Zielrechtecks.

Y: Bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Zielrechtecks.

Width: Bestimmt die Breite des Zielrechtecks in virtuellen Einheiten.

Height: Bestimmt die Höhe des Zielrechtecks in virtuellen Einheiten.

SrcDC: Bezeichnet den Gerätekontext, der das Quell-Bitmap enthält.

XSrc: Bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Quellrechtecks.

YSrc: Bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Quellrechtecks.

SrcWidth: Bestimmt die Breite des Quellrechtecks in virtuellen Einheiten.

SrcHeight: Bestimmt die Höhe des Quellrechtecks in virtuellen Einheiten.

Rop: Bestimmt die durchzuführende Rasteroperation. Die Codes der Rasteroperationen bestimmen, wie die GDI Farben bei Ausgabeoperationen kombiniert, die einen aktiven Pinsel, ein mögliches Quell-Bitmap und ein Ziel-Bitmap einschließen. Eine Liste der Codes für Rasteroperationen finden Sie in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »Ternäre Rasteroperationen«.

Rückgabewert Der Rückgabewert bestimmt, ob das Bitmap gezeichnet wurde. Er ist ungleich Null, wenn das Bitmap gezeichnet wurde, andernfalls ist er Null.

Hinweis *StretchBlt* streckt oder staucht das Quell-Bitmap im Speicher und kopiert dann das Ergebnis in das Ziel. Soll ein Muster mit dem Ergebnis verschmolzen werden, wird es solange nicht verschmolzen, bis das gestreckte Quell-Bitmap in das Ziel kopiert wird. Wird ein Pinsel verwendet, ist es der ausgewählte Pinsel im Zielgerätekontext. Die Zielkoordinaten werden in Übereinstimmung mit dem Zielgerätekontext, die Quellkoordinaten in Übereinstimmung mit dem Quellgerätekontext umgewandelt. Wenn Ziel-, Quell- und Muster-Bitmaps nicht das gleiche Farbformat haben, konvertiert *StretchBlt* das Quell- und das Muster-Bitmap so, daß sie zum Ziel-Bitmap passen. Die Vorder- und Hintergrundfarben des Ziels werden bei der Konvertierung verwendet.

Muß *StretchBlt* ein Monochrom-Bitmap in Farbe konvertieren, werden weiße Bits (1) auf die Hintergrundfarbe und schwarze Bits (0) auf die Vordergrundfarbe gesetzt. Um Farbe in Monochrom zu konvertieren, werden Pixel, die der Hintergrundfarbe entsprechen, auf weiß (1) und alle anderen Pixel auf schwarz (0) gesetzt. Bei Farbgeräten werden stattdessen die Vorder- und Hintergrundfarben des Gerätekontextes verwendet. Nicht alle Geräte unterstützen die Funktion *StretchBlt*.

Querverweis *SetStretchBltMode*

StretchDIBits

Syntax

```
int StretchDIBits(HDC DC, WORD DestX, WORD DestY,
WORD DestWidth, WORD DestHeight, WORD SrcX, WORD SrcY,
WORD SrcWidth, WORD SrcHeight, LPSTR Bits,
LPBITMAPINFO BitsInfo, WORD Usage, DWORD Rop);
```

Beschreibung

Diese Funktion bewegt ein geräteunbhängiges Bitmap (device-independent bitmap, DIB) aus einem Quell- in ein Zielrechteck und streckt oder staucht das Bitmap, wenn dies nötig ist, um es den Abmessungen des Zielrechtecks anzupassen. *StretchDIBits* verwendet den Streckmodus des Zielgerätekontextes (gesetzt durch die Funktion *SetStretchBltMode*), um zu bestimmen, wie das Bitmap zu strecken oder zu stauchen ist.

StretchDIBits bewegt das Bitmap von dem durch die Parameter *Bits*, *BitsInfo* und *Usage* festgelegten geräteunabhängigen Bitmap zu dem durch den Parameter *DC* bezeichneten Zielgerät. Die Parameter *SrcX*, *SrcY*, *SrcWidth* und *SrcHeight* definieren den Ursprung und die Abmessungen des Quellrechtecks. Der Ursprung des Koordinatensystems des geräteunabhängigen Bitmaps ist die untere linke Ecke. Die Parameter *DestX*, *DestY*, *DestWidth* und *DestHeight* geben den Ursprung und die Abmessungen des Zielrechtecks an. Der Koordinatenursprung des Ziels hängt vom aktuellen Abbildungsmodus des Gerätekontextes ab. Weitere Informationen zu den Abbildungsmodi finden sich im Abschnitt »MM_XXXX« in Kapitel 4 »Windows-Stile und -Konstanten«.

Die durch den Parameter *Rop* festgelegte Rasteroperation bestimmt, wie das Quell-Bitmap und die gegebenen Bits des Zielgerätes kombiniert werden.

StretchBlt erzeugt ein Spiegelbild des Bitmaps, wenn die Vorzeichen der Parameter *SrcWidth* und *DestWidth* oder *SrcHeight* und *DestHeight* voneinander abweichen.

Haben *SrcWidth* und *DestWidth* unterschiedliche Vorzeichen, erzeugt die Funktion ein Spiegelbild des Bitmaps entlang der x-Achse, haben *SrcHeight* und *DestHeight* unterschiedliche Vorzeichen, erzeugt die Funktion ein Spiegelbild des Bitmaps entlang der y-Achse.

DC: Bezeichnet den Zielgerätekontext für eine Bildschirmoberfläche oder ein Speicher-Bitmap.

DestX, DestY: Bestimmt die x- bzw. y-Koordinate des Ursprungs des Zielrechtecks in virtuellen Einheiten.

DestWidth, DestHeigth: Bestimmt die Breite bzw. Höhe des Zielrechtecks in virtuellen Einheiten.

SrcX, SrcY: Bestimmt die x- bzw. y-Koordinate des Quellrechtecks im DIB in Pixel.

SrcWidth, SrcHeigth: Bestimmt die Breite bzw. Höhe des Quellrechtecks im DIB in Pixel.

Bits: Bits des DIB, die in einem Array von Bytes abgelegt sind.

BitsInfo: Zeigt auf eine *BITMAPINFO*-Datenstruktur, die Informationen über das DIB enthält.

Usage: Bestimmt, ob die Felder *bmiColors*[*i*] in *BitsInfo* explizite RGB-Werte oder Indizes auf die gegenwärtig verwendete virtuelle Palette enthalten. Dieser Parameter muß einen der folgenden Werte annehmen:

Wert	Bedeutung
DIB_PAL_COLORS	Die Farbtabelle besteht aus einem Array mit 16-Bit-Indizes auf die aktuell verwendete virtuelle Palette.
DIB_RGB_COLORS	Die Farbtabelle enthält RGB-Werte.

Rop: Legt die durchzuführende Rasteroperation fest. Die Codes der Rasteroperationen bestimmen, wie die GDI Farben bei Ausgabeoperationen kombiniert, die einen aktiven Pinsel, ein mögliches Quell-Bitmap und ein Ziel-Bitmap einschließen.

Weitere Informationen finden sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »Ternäre Rasteroperationen«.

Rückgabewert Der Rückgabewert bestimmt die Anzahl der kopierten Scan-Zeilen.

SwapMouseButton

Syntax
```
BOOL SwapMouseButton(BOOL Swap);
```

Beschreibung Diese Funktion tauscht die Bedeutung von linker und rechter Maustaste aus. Ist der Parameter *Swap* ungleich Null, erzeugt die linke Taste Maussignale der rechten Taste und die rechte Taste Maussignale der linken Taste. Ist *Swap* Null, wird die ursprüngliche Bedeutung der Tasten wiederhergestellt.

Swap: Bestimmt, ob die Bedeutung der Tasten umgekehrt oder die ursprüngliche Bedeutung der Tasten wiederhergestellt werden soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion die Bedeutung der Maustasten vertauscht hat, andernfalls ist er Null.

Hinweis Das Umschalten der Tasten ist als Erleichterung für Linkshänder gedacht. Die Funktion *SwapMouseButton* wird gewöhnlich nur von der Systemsteuerung aufgerufen.

Obwohl es Anwendungen freisteht, die Funktion aufzurufen, ist die Maus eine gemeinsam genutzte Ressource. Das Vertauschen der Bedeutung der Maustasten beeinflußt also alle Anwendungen.

SwapRecording

Syntax
```
void SwapRecording(WORD Flag);
```

Beschreibung Wird mit Microsoft Windows Swap gearbeitet, startet oder beendet diese Funktion die Analyse des Auslagerungsverhaltens.

Flag: Bestimmt, ob Swap die Analyse starten oder beenden soll. Folgende Werte sind möglich:

Wert	Bedeutung
0	Beendet die Analyse.
1	Auslagerungsaufrufe und -rückgaben aufzeichnen.
2	Wie 1, zusätzlich werden Aufrufe über Thunks aufgezeichnet. Diese Option führt zur Aufzeichnung einer großen Datenmenge.

Rückgabewert Keiner.

SwitchStackBack

Syntax
```
void SwitchStackBack(void);
```

Beschreibung Diese Funktion gibt den Stack der aktuellen Task an das Datensegment der Task zurück, nachdem er vorher durch die Funktion *SwitchStackTo* neuadressiert wurde.

Die Funktion hat keine Parameter.

Rückgabewert Keiner.

Hinweis Diese Funktion sichert beim Rücksprung die Inhalte der AX:DX-Register.

Querverweis *SwitchStackTo*

SwitchStackTo

Syntax
```
void SwitchStackTo(WORD StackSegment,
WORD StackPointer, WORD StackTop);
```

Beschreibung Diese Funktion ändert den Stack der aktuellen Task auf das durch den Parameter *StackSegment* festgelegte Segment.

Dynamische Linkbibliotheken (DLL) haben keinen Stack. Stattdessen verwendet eine DLL den Stack der Task, die die

Bibliothek aufruft. Daher können DLL-Funktionen, die voraussetzen, daß die Inhalte des Code-Segment-Registers (CS) und des Stack-Segment-Registers identisch sind, nicht zum gewünschten Resultat führen. Die Funktion *SwitchStackTo* adressiert den Stack der Task auf das Datensegment einer DLL um und läßt zu, daß die DLL diese Funktionen aufruft. *SwitchStackTo* kopiert die Argumente der Task auf dem Stack auf die neue Position des Stacks.

StackSegment: Bestimmt das Datensegment, das den Stack enthalten soll.

StackPointer: Legt den Offset des Stack-Anfangs im Datensegment fest.

StackTop: Legt den Offset der Stack-Spitze relativ zum Stack-Anfang fest.

Rückgabewert Keiner.

Hinweis Eine Task kann *SwitchStackTo* aufrufen, bevor eine Funktion in einer DLL aufgerufen wird, die voraussetzt, daß die CS- und DS-Register gleich sind. Beim Rücksprung der DLL-Funktion muß die Task dann *SwitchStackBack* aufrufen, um ihren Stack auf ihr eigenes Datensegment umzuadressieren. Ebenso kann eine DLL *SwitchStackTo* aufrufen, bevor sie eine Routine aufruft, die die Identität von CS und DS voraussetzt, und kann dann vor der Rückkehr zu der Task, die die DLL-Funktion aufrief, *SwitchStackBack* aufrufen.

Aufrufe von *SwitchStackTo* und *SwitchStackBack* können nicht verschachtelt werden. Das bedeutet, daß nach dem Aufruf von *SwitchStackTo* ein Programm *SwitchStackBack* aufrufen muß, bevor *SwitchStackTo* erneut aufgerufen werden kann.

Querverweis *SwitchStackBack*

SyncAllVoices

Syntax `int SyncAllVoices(void);`

Beschreibung Diese Funktion reiht eine Synchronisationsmarkierung in jede Klangwarteschlange ein. Nachdem in einer Klangwarteschlange eine Synchronisationsmarkierung erkannt wurde, wird die Ausgabe abgeschaltet, bis Synchronisationsmarkierungen in allen anderen Warteschlangen aufgefunden werden. Dies bewirkt die Synchronisation aller Tonausgaben.

Die Funktion hat keine Parameter.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war. Ist eine Klangwarteschlange voll, ist der Rückgabewert *S_SERQFUL*.

TabbedTextOut

Syntax

```
LONG TabbedTextOut(HDC DC, int X, int Y, LPSTR Str,
int Count, int TabPositions, LPINT TabStopPositions,
int TabOrigin);
```

Beschreibung

Diese Funktion schreibt einen String auf den angegebenen Bildschirm unter Verwendung der aktuell ausgewählten Schrift und erweitert dabei Tabulator-Zeichen zu den im Array *TabStopPositions* angegebenen Spaltenpositionen.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate des Startpunkts des Strings.

Y: Bestimmt die virtuelle y-Koordinate des Startpunkts des Strings.

Str: Zeigt auf den String, der gezeichnet werden soll.

Count: Bestimmt die Anzahl der Zeichen im String (ohne Nullzeichen). Ist *Count* größer als die tatsächliche Länge des Strings, stoppt *TabbedTextOut* die Ausgabe *nicht*, wenn es auf ein Nullzeichen trifft.

TabPositions: Bestimmt die Anzahl der Tabulatorpositionen im Array *TabStopPositions*.

TabStopPositions: Zeigt auf ein Array von Integerzahlen, das die Tabulatorpositionen in Pixel enthält. Die Tabulatorstops müssen in aufsteigender Reihenfolge sortiert werden; Rückwärtstabulatoren sind nicht zulässig.

TabOrigin: Bestimmt die virtuelle x-Koordinate der Startposition, von der aus die Tabulatoren erweitert werden.

Rückgabewert

Der Rückgabewert bestimmt die Dimensionen des Strings. Die Höhe ist im höherwertigen, die Breite im niederwertigen WORD enthalten.

Hinweis

Wenn die Parameter *TabPositions* und *TabStopPositions* 0 sind, werden die Tabulatoren auf acht durchschnittliche Zeichenbreiten erweitert.

Wenn *TabPositions* 1 ist, werden die Tabulatorstops in dem Abstand verteilt, den der erste Wert im Array *TabStopPositions* bestimmt.

Wenn *TabStopPositions* mehr als einen Wert enthält, wird ein Tabulatorstop für jeden Wert im Array gesetzt, bis zu der Zahl, die *TabPositions* angibt.

Der Parameter *TabOrigin* gestattet es einer Anwendung, die Funktion *TabbedTextOut* mehrmals für eine Zeile aufzurufen. Wenn die Anwendung *TabbedTextOut* mehr als einmal aufruft und dabei *TabOrigin* jedesmal auf denselben Wert gesetzt ist,

erweitert die Funktion alle Tabulatoren relativ zu der Position, die durch *TabOrigin* vorgegeben ist.

TextOut

Syntax

```
BOOL TextOut(HDC DC, int X, int Y, LPSTR Str, int
Count);
```

Beschreibung Diese Funktion schreibt einen String unter Verwendung der aktuell ausgewählten Schrift auf den bezeichneten Bildschirm. Die Startposition des Strings ist durch die Parameter *X* und Y gegeben.

DC: Bezeichnet den Gerätekontext.

X: Bestimmt die virtuelle x-Koordinate des Startpunkts des Strings.

Y: Bestimmt die virtuelle y-Koordinate des Startpunkts des Strings.

Str: Zeigt auf den String, der gezeichnet werden soll.

Count: Bestimmt die Anzahl der Zeichen im String (ohne Nullzeichen). Ist *Count* größer als die tatsächliche Länge des Strings, stoppt *TabbedTextOut* die Ausgabe *nicht*, wenn es auf ein Nullzeichen trifft.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn der String gezeichnet wurde, andernfalls ist er Null.

Hinweis Als Ursprung eines Zeichens ist die obere linke Ecke der Zeichenzelle definiert. Gemäß Voreinstellung wird die aktuelle Position von dieser Funktion weder verwendet noch aktualisiert. Eine Anwendung kann aber *SetTextAlign* mit auf *TA_UPDATECP* gesetztem Parameter *Flags* aufrufen, um es Windows zu ermöglichen, die aktuelle Position jedesmal zu verwenden und zu aktualisieren, wenn die Anwendung *TextOut* für einen gegebenen Gerätekontext aufruft.

Wenn dieses Flag gesetzt ist, ignoriert Windows die Parameter *X* und *Y* nachfolgender Aufrufe von *TextOut*.

Querverweis *SetTextAlign*

Throw

Syntax

```
void Throw(LPCATCHBUF CatchBuf, int ThrowBack);
```

Beschreibung Diese Funktion stellt die Ausführungsumgebung mit den Werten wieder her, die in *CatchBuf* gespeichert sind.

Die Ausführungsumgebung ist der Status aller Systemregi-

ster und des Befehlszählers. Die Ausführung fährt mit der Funktion *Catch* fort, die die Umgebung kopiert hat.

Der Parameter *ThrowBack* wird als Rückgabewert an die Funktion *Catch* übergeben und kann ungleich 0 sein.

CatchBuf: Ein Array, das die Ausführungsumgebung enthält.

ThrowBack: Bestimmt den Wert, der an die Funktion *Catch* übergeben werden soll.

Rückgabewert Keiner.

ToAscii

Syntax

```
int ToAscii(WORD VirtKey, WORD ScanCode,
LPSTR KeyState, LPVOID CharBuff, WORD Flags);
```

Beschreibung Diese Funktion übersetzt den durch den Parameter *VirtKey* bestimmten virtuellen Tastencode und den durch den Parameter *KeyState* bezeichneten aktuellen Tastaturstatus in das oder die zugehörige(n) ANSI-Zeichen.

VirtKey: Bestimmt den virtuellen Tastencode, der übersetzt werden soll.

ScanCode: Bestimmt den ursprünglichen Hardware-Scancode der Taste, die übersetzt werden soll. Das höchstwertige Bit dieses Werts wird gesetzt, wenn die Taste nicht gedrückt ist.

KeyState: Zeigt auf ein Array von 256 Bytes, von denen jedes den Status einer Taste enthält. Wenn das höchstwertige Bit des jeweiligen Bytes gesetzt ist, ist die Taste nicht gedrückt.

CharBuff: Zeigt auf einen 32-Bit-Puffer, der das oder die übersetzten ANSI-Zeichen aufnimmt.

Flags: Das Bit 0 wird intern als Flag für die Menüanzeige verwendet.

Rückgabewert Der Rückgabewert bestimmt die Zahl der Zeichen, die in den Puffer kopiert wurden, auf den der Parameter *CharBuff* zeigt. Der Rückgabewert ist negativ, wenn die Taste eine tote Taste war, andernfalls ist er einer der folgenden Werte:

Wert	Bedeutung
2	Es wurden zwei Zeichen in den Puffer kopiert. Das sind gewöhnlich ein Akzent und das Zeichen einer tote Taste, falls die tote Taste nicht anders übersetzt werden kann.
1	Es wurde ein ANSI-Zeichen in den Puffer kopiert.
0	Die angegebene virtuelle Taste hat für den aktuellen Tastaturstatus keine Übersetzung.

Hinweis

Die Argumente, die an die Funktion *ToAscii* übergeben werden, reichen eventuell nicht aus, den virtuellen Tastencode zu übersetzen, weil vorher eine tote Taste im Tastaturtreiber gespeichert wurde. Üblicherweise führt *ToAscii* die Übersetzung gemäß des virtuellen Tastencodes durch. In einigen Fällen wird jedoch der Parameter *ScanCode* verwendet, um zwischen dem Drücken und dem Loslassen einer Taste zu unterscheiden. Der Scancode wird für die Übersetzung der Kombinationen von Alt mit numerischen Tasten verwendet.

TrackPopupMenu

Syntax

```
BOOL TrackPopupMenu(HMENU Menu, WORD Flags, int x,
int y, int cx, HWND Wnd, LPRECT Rect);
```

Beschreibung

Diese Funktion stellt ein Pop-up-Menü an der angegebenen Position dar und ermittelt die Tracking-Auswahl der Optionen aus dem Pop-up-Menü.

Ein Pop-up-Menü für Tracking-Auswahl kann irgendwo auf dem Bildschirm erscheinen. Der Parameter *Menu* bezeichnet das Handle des darzustellenden Menüs. Die Funktion liefert dieses Handle durch Aufruf von *CreatePopupMenu*, um ein neues Pop-up-Menü anzulegen oder durch Aufruf von *GetSubMenu*, um das Handle eines Pop-up-Menüs, das einer vorhandenen Menüoption zugeordnet ist, wiederzuerhalten. Windows sendet Botschaften, die vom Menü erzeugt wurden, an das durch den Parameter *Wnd* bezeichnete Fenster.

Menu: Bezeichnet das Pop-up-Menü, das dargestellt werden soll.

Flags: Wird nicht verwendet. Dieser Parameter muß auf 0 gesetzt werden.

x: Bestimmt die horizontale Position der linken Seite des Menüs auf dem Bildschirm in Bildschirmkoordinaten.

y: Bestimmt die vertikale Position der Oberkante des Menüs auf dem Bildschirm in Bildschirmkoordinaten.

cx: Bestimmt die Breite des Menüs in Bildschirmeinheiten. Ist *cx* Null, wird die Voreinstellung verwendet.

Wnd: Bezeichnet das Fenster, welches das Pop-up-Menu besitzt. Dieses Fenster empfängt alle *WM_COMMAND*- Botschaften vom Menü.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die festlegt, in welchem Bereich sich der Mauszeiger befinden muß, damit das Menü sichtbar bleibt, wenn der Anwender die Maustaste losläßt.

Rückgabewert Der Rückgabewert bestimmt das Ergebnis der Funktion. Er ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Querverweis *CreatePopupMenu, GetSubMenu*

TranslateAccelerator

Syntax
```
int TranslateAccelerator(HWND Wnd, HANDLE AccTable,
LPMSG Msg);
```

Beschreibung Diese Funktion bearbeitet Akzelerator-Tasten für Menübefehle. *TranslateAccelerator* übersetzt die *WM_KEYUP-* und *WM_KEYDOWN*-Botschaften in *WM_COMMAND-* oder *WM_SYSCOMMAND*-Botschaften, wenn in der Akzelerator-Tabelle der Anwendung ein Eintrag für die Taste vorhanden ist. Das höherwertige WORD des Parameters *lParam* der Botschaft *WM_COMMAND* oder *WM_SYSCOMMAND* enthält den Wert 1, um diese Botschaft von anderen Botschaften, die von Menüs oder Steuerelementen gesendet werden, zu unterscheiden. Die Botschaften *WM_COMMAND* oder *WM_SYSCOMMAND* werden direkt an das Fenster gesendet und nicht in die Anwendungswarteschlange gestellt. Die Funktion *TranslateAccelerator* kehrt nicht zurück, bis die Botschaft bearbeitet ist. Akzelerator-Tasten, die dazu definiert sind, Optionen aus dem Systemmenü auszuwählen, werden in *WM_SYSCOMMAND-*, alle anderen in *WM_COMMAND*-Botschaften übersetzt.

Wnd: Bezeichnet das Fenster, dessen Botschaften übersetzt werden sollen.

AccTable: Bezeichnet eine Akzelerator-Tabelle, die mit der Funktion *LoadAccelerators* geladen wurde.

Msg: Zeigt auf eine *MSG*-Struktur mit Botschaftsinformationen, die mit *PeekMessage* oder *GetMessage* geliefert werden.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn die Übersetzung stattfindet, andernfalls ist er 0.

Hinweis Wenn *TranslateAccelerator* einen Wert ungleich 0 zurückgibt (die Botschaft also übersetzt ist), sollte die Anwendung die gleiche Botschaft nicht nochmals mit der Funktion *TranslateMessage* bearbeiten. Befehle in Akzelerator-Tabellen müssen keine Entsprechungen in den Menüoptionen haben.

Entspricht der Akzelerator-Befehl der Menüoption, werden der Anwendung *WM_INITMENU-* und *WM_INITMENUPOPUP*-Botschaften übergeben, so als ob der Benutzer versuchen würde, das Menü anzuzeigen. Diese Botschaften werden jedoch nicht gesendet, wenn eine der folgenden Bedingungen erfüllt ist:

- Das Fenster ist gesperrt.
- Die Menüoption ist gesperrt.
- Der Befehl ist nicht im Systemmenü und das Fenster ist verkleinert.
- Ein Mausfang ist aktiv (siehe auch Funktion *SetCapture*).

Wenn das Fenster das aktive Fenster ist und kein Tastaturfokus zugewiesen wurde (das trifft im allgemeinen zu, wenn das Fenster verkleinert ist), so werden *WM_SYSKEYUP*- und *WM_SYSKEYDOWN*-Botschaften anstelle von *WM_KEYUP*- und *WM_KEYDOWN*-Botschaften gesendet. Tritt eine einer Menüoption entsprechende Akzelerator-Eingabe auf, während das Fenster, welches das Menü besitzt, als Symbol dargestellt ist, wird keine *WM_COMMAND*-Botschaft gesendet. Wenn jedoch eine Akzelerator-Eingabe auftritt, die mit keiner der Optionen im Menü des Fensters übereinstimmt, so wird eine *WM_COMMAND*-Botschaft gesendet – auch dann, wenn das Fenster als Symbol dargestellt ist.

TranslateMDISysAccel

Syntax
```
BOOL TranslateMDISysAccel(HWND Wnd, LPMSG Msg);
```

Beschreibung Diese Funktion bearbeitet Akzelerator-Tasten für Systemmenübefehle untergeordneter Fenster von Multidokumentschnittstellen (MDI).

TranslateMDISysAccel übersetzt *WM_KEYUP*- und *WM_KEYDOWN*-Botschaften in *WM_SYSCOMMAND*-Botschaften. Dabei hat das höherwertige WORD des Parameters *lParam* der *WM_SYSCOMMAND*-Botschaft den Wert 1, um diese Botschaft von Botschaften zu unterscheiden, die von Menüs oder Steuerelementen gesendet werden.

WndClient: Bezeichnet das übergeordnete MDI-Client-Fenster.

Msg: Zeigt auf eine *MSG*-Struktur mit Botschaftsinformationen, die mit *PeekMessage* oder *GetMessage* geliefert werden.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion eine Botschaft in einen Systembefehl übersetzt hat, andernfalls ist er Null.

TranslateMessage

Syntax
```
BOOL TranslateMessage(LPMSG Msg);
```

Beschreibung Diese Funktion übersetzt virtuelle Tastenbotschaften in Zeichenbotschaften wie folgt:

WM_KEYDOWN/WM_KEYUP-Kombinationen produzieren eine Botschaft *WM_CHAR* oder *WM_DEADCHAR*.

WM_SYSKEYDOWN/WM_SYSKEYUP-Kombinationen produzieren eine Botschaft *WM_SYSCHAR* oder *WM_SYSDEADCHAR*.

Die Zeichenbotschaften werden in die Anwendungswarteschlange eingereiht, um beim nächsten Aufruf der Funktion *GetMessage* oder *PeekMessage* durch die Anwendung gelesen zu werden.

Msg: Zeigt auf eine *MSG*-Struktur mit Botschaftsinformationen, die mit einer der Funktionen *PeekMessage* oder *GetMessage* geliefert werden.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Botschaft übersetzt wurde (d.h. Zeichenbotschaften in die Anwendungswarteschlange gestellt werden), andernfalls ist er Null.

Hinweis *TranslateMessage* verändert die durch den Parameter *Msg* bezeichnete Botschaft nicht. *TranslateMessage* erzeugt *WM_CHAR*-Botschaften nur für die Tasten, denen durch den Tastaturtreiber ASCII-Zeichen zugeordnet sind.

Eine Anwendung sollte *TranslateMessage* dann nicht aufrufen, wenn die Anwendung virtuelle Tastenbotschaften für irgendwelche anderen Zwecke verarbeitet. Zum Beispiel sollte eine Anwendung *TranslateMessage* nicht aufrufen, wenn die Funktion *TranslateAccelerator* einen Wert ungleich 0 zurückgibt.

TransmitCommChar

Syntax `int TransmitCommChar(int Cid, char AChar);`

Beschreibung Diese Funktion markiert das durch den Parameter *AChar* angegebene Zeichen zur sofortigen Übertragung, indem sie es an den Anfang der Übertragungswarteschlange stellt.

Cid: Bestimmt die Schnittstelle, die das Zeichen empfangen soll. Die Funktion *OpenComm* gibt diesen Wert zurück.

AChar: Bestimmt das zu übertragende Zeichen.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war und negativ, wenn das Zeichen nicht übertragen werden kann. Ein Zeichen kann nicht übertragen werden, wenn das durch den vorherigen Aufruf der Funktion *TransmitCommChar* bestimmte Zeichen noch nicht gesendet wurde.

Querverweis *OpenComm*

UngetCommChar

Syntax
```
int UngetCommChar(int Cid, char AChar);
```

Beschreibung Diese Funktion stellt das durch den Parameter *AChar* bestimmte Zeichen in die Empfangswarteschlange zurück und macht es zu dem Zeichen, das bei einem darauffolgenden Lesevorgang als erstes aus der Warteschlange gelesen wird.

Nachfolgende Aufrufe der Funktion *UngetCommChar* sind nicht erlaubt. Das Zeichen, das in die Warteschlange zurückgestellt wurde, muß vor dem Versuch, ein weiteres Zeichen zurückzustellen, gelesen werden.

Cid: Bestimmt das Datenübertragungsgerät, welches das Zeichen empfangen soll.

AChar: Bestimmt das Zeichen, das in die Empfangswarteschlange zurückgestellt werden soll.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war und negativ, wenn ein Fehler auftritt.

Querverweis *OpenComm*

UnhookWindowsHook

Syntax
```
BOOL UnhookWindowsHook(int Hook, FARPROC HookFunc);
```

Beschreibung Diese Funktion entfernt die Windows-Filterverwaltungsfunktion, die der Parameter *Hook* bestimmt, aus einer Kette von Filterverwaltungsfunktionen.

Eine Windows-Filterverwaltungsfunktion verarbeitet Ereignisse, bevor diese an die Botschaftsschleife einer Anwendung in der Funktion *WinMain* gesendet wird.

Hook: Bestimmt den Typ der zu entfernenden Filterverwaltungsfunktion. Er kann einen der folgenden Werte annehmen.

Wert	Bedeutung
WH_CALLWNDPROC	Entfernt einen Fensterfunktionsfilter.
WH_GETMESSAGE	Entfernt einen Botschaftsfilter.
WH_JOURNALPLAYBACK	Entfernt einen Filter für ein Wiedergabeprotokoll.
WH_JOURNALRECORD	Entfernt einen Filter für ein Aufzeichnungsprotokoll.
WH_KEYBOARD	Entfernt einen Tastaturfilter.
WH_MSGFILTER	Entfernt einen Botschaftsfilter.

HookFunc: Ist die Adresse der Prozedurinstanz der Filterverwaltungsfunktion.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Filterverwaltungsfunktion erfolgreich entfernt wurde, andernfalls ist er Null.

UnionRect

Syntax
```
int UnionRect(LPRECT DestRect, LPRECT Src1Rect,
LPRECT Src2Rect);
```

Beschreibung Diese Funktion erzeugt die Vereinigung zweier Rechtecke. Die Vereinigung ist das kleinste Rechteck, das beide Quellrechtecke einschließt.

DestRect: Zeigt auf eine *RECT*-Datenstruktur, die das neue Rechteck aufnehmen soll.

Src1Rect: Zeigt auf eine *RECT*-Datenstruktur, die ein Quellrechteck enthält.

Src2Rect: Zeigt auf eine *RECT*-Datenstruktur, die ein Quellrechteck enthält.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn die Vereinigung nicht leer und 0, wenn die Vereinigung leer ist.

Hinweis Windows ignoriert die Abmessungen eines leeren Rechtecks (also eines Rechtecks, das keine Höhe oder keine Breite hat).

UnlockData

Syntax
```
HANDLE UnlockData(int Dummy);
```

Beschreibung Dieses Makro hebt die Sperre des aktuellen Datensegments auf. Es ist für die Verwendung durch Module vorgesehen, die verschiebbare Datensegmente haben.

Dummy: Wird nicht verwendet und sollte auf 0 gesetzt werden.

Rückgabewert Es wird der Bezeichner des entsperrten Segments zurückgegeben oder Null, wenn das Makro fehlschlägt.

UnlockResource

Syntax
```
BOOL UnlockResource(HANDLE ResData);
```

Beschreibung Dieses Makro hebt die Sperre der durch den Parameter *ResData* bezeichneten Ressource auf und verringert den Referenzzähler der Ressource um 1.

ResData: Bezeichnet den globalen Speicherblock, dessen Sperre aufgehoben werden soll.

Rückgabewert Der Rückgabewert ist 0, wenn der Referenzzähler des Blocks bis auf 0 herabgesetzt wurde, andernfalls ist er ungleich 0.

UnlockSegment

Syntax
```
HANDLE UnlockSegment(WORD Segment);
```

Beschreibung Diese Funktion hebt die Sperre des Segments auf, dessen Segmentadresse durch den Parameter *Segment* angegeben ist. Wenn *Segment* -1 ist, hebt die Funktion *UnlockSegment* die Sperre des aktuellen Datensegments auf.

Im Real Mode oder wenn das Segment das Attribut »discardable« hat, verringert *UnlockSegment* den Sperrenzähler des Segments um 1. Im Protected Mode verringert *UnlockSegment* nur den Sperrenzähler von Objekten mit dem Attribut »discardable« und von automatischen Datensegmenten. Wenn der Sperrenzähler auf 0 zurückgesetzt ist, wurde die Sperre des Segments vollständig aufgehoben und das Segment kann verschoben oder verworfen werden.

Andere Funktionen können den Sperrenzähler eines Speicherobjektes ebenfalls beeinflussen. In der Beschreibung der Funktion *GlobalFlags* findet sich eine Liste dieser Funktionen.

Eine Anwendung muß jeden Aufruf von *LockSegment* für ein Segment mit einem entsprechenden *UnlockSegment* wieder aufheben.

Segment: Bestimmt die Segmentadresse des Segments, dessen Sperre aufgehoben werden soll. Wenn *Segment* -1 ist, hebt *UnlockSegment* die Sperre des aktuellen Datensegments auf.

Rückgabewert Der Rückgabewert ist 0, wenn der Sperrenzähler des Segments auf 0 heruntergesetzt wurde. Andernfalls ist der Rückgabewert ungleich 0.

Eine Anwendung sollte sich nicht auf den Rückgabewert verlassen, um zu bestimmen, wie oft nacheinander sie *UnlockSegment* für das Segment aufrufen muß, um es vollständig zu entsperren.

Querverweis *LockSegment*

UnrealizeObject

Syntax
```
BOOL UnrealizeObject(HBRUSH HObject);
```

Beschreibung Bezeichnet der Parameter *HObject* einen Pinsel, weist diese Funktion die GDI an, den Ursprung dieses Pinsels zurückzusetzen, wenn er das nächste Mal ausgewählt wird. Bezeichnet *HObject* eine virtuelle Palette, weist diese Funktion die GDI an, eine Palette zu realisieren, als wäre sie vorher noch nicht realisiert worden. Ruft die Anwendung das nächste Mal die

Funktion *RealizePalette* für die angegebene Palette auf, legt die GDI die virtuelle Palette in der Systempalette komplett neu an.

HObject: Bezeichnet das Objekt, das zurückgesetzt werden soll.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Die Funktion *UnrealizeObject* sollte nicht für vordefinierte Objekte verwendet werden. Diese Funktion muß immer aufgerufen werden, wenn ein neuer Pinselursprung mit der Funktion *SetBrushOrg* gesetzt wird. Ein Pinsel, der durch den Parameter *HObject* bezeichnet wird, darf nicht der aktuell ausgewählte Pinsel eines Bildschirmkontextes sein. Eine Palette, die durch *HObject* bezeichnet ist, darf die aktuell ausgewählte Palette eines Bildschirmkontextes sein.

UnregisterClass

Syntax
```
BOOL UnregisterClass(LPSTR ClassName, HANDLE Instance);
```

Beschreibung Diese Funktion löscht die die durch *ClassName* bezeichnete Fensterklasse aus der Fensterklassentabelle und gibt dabei den der Klasse zugewiesenen Speicher frei.

ClassName: Zeigt auf einen nullterminierten String, der die Klassenbezeichnung enthält. Diese Klassenbezeichnung muß vorher durch Aufruf der Funktion *RegisterClass* mit einem gültigen Feld *hInstance* in der *WNDCLASS*-Struktur registriert werden. Vordefinierte Klassen, wie beispielsweise die von Dialogfenstersteuerelementen, können nicht gelöscht werden.

Instance: Bezeichnet die Instanz des Moduls, das die Klasse erzeugt hat.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion die Fensterklasse erfolgreich aus der Fensterklassentabelle gelöscht hat. Er ist Null, wenn die Klasse nicht gefunden wurde oder wenn ein Fenster existiert, das mit dieser Klasse erzeugt wurde.

Hinweis Bevor diese Funktion benutzt wird, müssen alle Fenster, die mit der angegebenen Klasse erzeugt wurden, entfernt und freigegeben werden.

Querverweis *RegisterClass*

UpdateColors

Syntax
```
int UpdateColors(HDC DC);
```

Beschreibung Diese Funktion aktualisiert den Client-Bereich des durch den Parameter *DC* bezeichneten Gerätekontextes, indem die aktu-

ellen Farben im Client-Bereich pixelweise an die Systempalette angeglichen werden. Ein inaktives Fenster mit einer realisierten virtuellen Palette kann *UpdateColors* als Alternative zum Neuzeichnen seines Client-Bereiches aufrufen, wenn sich die Systempalette ändert.

DC: Bezeichnet den Gerätekontext.

Rückgabewert Der Rückgabewert wird nicht verwendet.

Hinweis *UpdateColors* aktualisiert einen Client-Bereich normalerweise schneller, als er neu gezeichnet werden kann. Da *UpdateColors* jedoch die Farbumwandlung gemäß der Farbe eines jeden Pixels vor der Änderung der Systempalette ausführt, bewirkt jeder Aufruf dieser Funktion einen gewissen Verlust an Farbwiedergabetreue.

UpdateWindow

Syntax `void UpdateWindow(HWND Wnd);`

Beschreibung Diese Funktion aktualisiert den Client-Bereich des gegebenen Fensters, indem sie eine *WM_PAINT*-Botschaft an das Fenster sendet, wenn der Aktualisierungsbereich für das Fenster nicht leer ist.

Die Funktion *UpdateWindow* sendet eine *WM_PAINT*- Botschaft unter Umgehung der Anwendungs-Warteschlange direkt an die Fensterfunktion des gegebenen Fensters. Ist die Aktualisierungsregion leer, wird keine Botschaft gesendet.

Wnd: Bezeichnet das zu aktualisierende Fenster.

Rückgabewert Keiner.

ValidateCodeSegments

Syntax `void ValidateCodeSegments(void);`

Beschreibung Diese Funktion gibt Debugging-Informationen auf ein Terminal aus, wenn Codesegmente durch zufällige Speicherüberschreibungen verändert wurden. Die Funktion ist nur in der Debugging-Version von Windows verfügbar und standardmäßig aktiviert. Um sie zu deaktivieren, muß das Flag *EnableSegmentChecksum* im Abschnitt [kernel] der Datei WIN.INI auf 0 gesetzt werden. Im Protected Mode (Standard- oder erweiterter 386-Modus) überprüft Windows die Gültigkeit der Codesegmente nicht.

Die Funktion hat keine Parameter.

Rückgabewert Keiner.

ValidateFreeSpaces

Syntax
```
LPSTR ValidateFreeSpaces(void);
```

Beschreibung Diese Funktion ist nur in der Debugging-Version von Windows verfügbar. Sie überprüft freie Segmente auf gültige Inhalte. In der Debugging-Version von Windows füllt das Kernel alle Bytes in freien Segmenten mit dem Hexadezimalwert 0xCC.

Diese Funktion beginnt mit der Überprüfung im freien Segment mit der niedrigsten Adresse und fährt solange fort, bis ein ungültiges Byte gefunden wird oder die Überprüfung ergibt, daß der gesamte freie Speicher gültige Inhalte hat. Vor Aufruf dieser Funktion müssen Sie die Datei WIN.INI um folgende Zeilen ergänzen:

```
[kernel]
EnableFreeChecking=1
EnableHeapChecking=1
```

Die Funktion hat keine Parameter.

Rückgabewert Keiner.

Hinweis Windows übergibt die Debugging-Informationen an den Debugger, wenn ein ungültiges Byte gefunden wird. Anschließend wird ein *Fatal Exit* ausgeführt. Die [kernel]-Einträge in WIN.INI sorgen für die automatische Überprüfung des freien Speichers.

Bevor ein Speicherblock als Reaktion auf einen Aufruf der Funktion *GlobalAlloc* zurückgegeben wird, überprüft Windows diesen Speicher, um sicherzustellen, daß er mit 0xCC gefüllt ist. Vor Aufruf der Funktion *GlobalCompact* wird der gesamte freie Speicher überprüft.

Der Aufruf dieser Funktion verlangsamt Windows um etwa 20%.

ValidateRect

Syntax
```
void ValidateRect(HWND Wnd, LPRECT Rect);
```

Beschreibung Diese Funktion führt eine Gültigkeitsüberprüfung des Client-Bereichs innerhalb des gegebenen Rechtecks durch, indem sie das Rechteck aus der Aktualisierungsregion des angegebenen Fensters löscht. Wenn der Parameter *Rect* NULL ist, wird das gesamte Fenster auf Gültigkeit überprüft.

Wnd: Bezeichnet das Fenster, dessen Aktualisierungsregion verändert werden soll.

Rect: Zeigt auf eine *RECT*-Datenstruktur, die das Rechteck, das aus der Aktualisierungsregion gelöscht werden soll, in Client-Koordinaten enthält.

Rückgabewert Keiner.

Hinweis Die Funktion *BeginPaint* führt automatisch eine Gültigkeitsprüfung für den gesamten Client-Bereich durch. Wenn nur ein Teil der Aktualisierungsregion auf Gültigkeit überprüft werden muß, sollte keine der Funktionen *ValidateRect* und *ValidateRgn* aufgerufen werden, bevor die nächste Botschaft *WM_PAINT* erzeugt wird.

Windows erzeugt solange *WM_PAINT*-Botschaften, bis die Gültigkeitsüberprüfung der gegenwärtigen Aktualisierungsregion durchgeführt ist.

Querverweis *BeginPaint*

ValidateRgn

Syntax
```
void ValidateRgn(HWND Wnd, HRGN Rgn);
```

Beschreibung Diese Funktion führt eine Gültigkeitsüberprüfung des Client-Bereichs innerhalb der bezeichneten Region durch, indem sie die Region aus der Aktualisierungsregion des angegebenen Fensters löscht.

Ist der Parameter *Rgn* 0, wird das gesamte Fenster auf Gültigkeit überprüft.

Wnd: Bezeichnet das Fenster, dessen Aktualisierungsregion verändert werden soll.

Rgn: Bezeichnet eine Region, die einen Bereich definiert, der aus der Aktualisierungsregion gelöscht werden soll.

Rückgabewert Keiner.

Hinweis Die angegebene Region muß vorher mit einer Region-Funktion erstellt worden sein. Die Koordinaten der Region müssen als Client-Koordinaten vorliegen.

VkKeyScan

Syntax
```
WORD VkKeyScan (WORD AChar);
```

Beschreibung Diese Funktion übersetzt ein ANSI-Zeichen in den zugehörigen virtuellen Tastencode und Shift-Status, bezogen auf die aktuelle Tastatur. *VkKeyScan* wird von Anwendungen verwendet, die Zeichen mit den Botschaften *WM_KEYUP* und *WM_KEYDOWN* senden.

AChar: Bestimmt das Zeichen, für das der zugehörige virtuelle Tastencode gefunden werden soll.

Rückgabewert Der virtuelle Tastencode wird im niederwertigen Byte zurückgegeben, der Shift-Status im höherwertigen Byte.

Mögliche Werte sind:

Wert	Bedeutung
0	Kein Shift.
1	Zeichentaste ist zusammen mit der Shift-Taste gedrückt.
2	Zeichen ist ein Steuerzeichen.
3, 4, 5	Eine Shift-Tastenkombination, die nicht für Zeichen benutzt wird.
6	Zeichen ist Ctrl+Alt.
7	Zeichen ist Shift+Ctrl+Alt.

Wird keine Taste gefunden, deren Übersetzung dem übergebenen ANSI-Code entspricht, wird sowohl im höherwertigen als auch im niederwertigen Byte -1 zurückgegeben.

Hinweis Übersetzungen für das numerische Tastenfeld (*VK_NUMPAD0* bis *VK_DIVIDE*) werden ignoriert. Diese Funktion ist dafür vorgesehen, nur eine Übersetzung für die Haupttastatur vorzunehmen.

WaitMessage

Syntax
```
void WaitMessage(void);
```

Beschreibung Diese Funktion gibt die Programmsteuerung an andere Anwendungen weiter, wenn eine Anwendung keine anderen Aufgaben auszuführen hat.

WaitMessage unterbricht die Anwendung und kehrt solange nicht zurück, bis eine neue Botschaft in die Anwendungswarteschlange gestellt wird.

Die Funktion hat keine Parameter.

Rückgabewert Keiner.

Hinweis *GetMessage*, *PeekMessage* und *WaitMessage* übergeben die Programmsteuerung an andere Anwendungen. Aufrufe dieser Funktionen sind die einzige Möglichkeit, andere Anwendungen ablaufen zu lassen.

Wenn *GetMessage*, *PeekMessage* und *WaitMessage* die Programmsteuerung an andere Anwendungen abgeben, können sich der Stack und die Datensegmente der die Funktion aufrufenden Anwendung im Arbeitsspeicher verschieben, um sich veränderten Speicheranforderungen anderer Anwendungen anzupassen. Hält die Anwendung Zeiger auf Objekte im Daten- oder Stack-Bereich gespeichert (d.h. globale oder lokale Variablen), können diese Zeiger nach einem Aufruf von *GetMessage*, *PeekMessage* oder *WaitMessage* ungültig werden.

WaitSoundState

Syntax
```
int WaitSoundState(int State);
```

Beschreibung Diese Funktion wartet, bis der Klangtreiber den angegebenen Zustand erreicht hat.

State: Bestimmt den Zustand der Klangwarteschlangen. Er kann einen der folgenden Werte annehmen:

Wert	Bedeutung
S_ALLTHRESHLOD	Alle Klangwarteschlangen haben den Schwellenwert erreicht.
S_QUEUEEMPRTY	Alle Klangwarteschlangen sind leer und die Klangtreiber ausgeschaltet.
S_THRESHOLD	Eine Klangwarteschlange hat den Schwellenwert erreicht und gibt den Klang aus.

Rückgabewert Der Rückgabewert ist 0, wenn die Funktion erfolgreich war. Wenn der Zustand ungültig ist, ist er *S_SERDST*.

WindowFromPoint

Syntax
```
HWND WindowFromPoint(POINT Point);
```

Beschreibung Diese Funktion ermittelt das Fenster, das den vorgegebenen Punkt enthält. Der Parameter *Point* muß die Bildschirmkoordinaten eines Punktes auf dem Bildschirm bestimmen.

Point: Eine *POINT*-Datenstruktur, die den zu untersuchenden Punkt enthält.

Rückgabewert Der Rückgabewert bezeichnet das Fenster, in dem der Punkt liegt. Er ist 0, wenn an dem angegebenen Punkt kein Fenster existiert.

WinExec

Syntax
```
WORD WinExec(LPSTR CmdLine, WORD CmdShow);
```

Beschreibung Diese Funktion bringt die durch den Parameter *CmdLine* bezeichnete Windows- oder Nicht-Windows-Anwendung zur Ausführung. *CmdShow* bestimmt den Anfangszustand des Hauptfensters der Anwendung, wenn es erstellt wird.

CmdLine: Zeigt auf einen nullterminierten String, der die Befehlszeile für die auszuführende Anwendung enthält (Dateiname und optionale Parameter). Enthält der String *CmdLine* keinen Verzeichnispfad, sucht Windows in folgender Reihenfolge nach der ausführbaren Datei:

1. Im aktuellen Verzeichnis

2. Im Windows-Verzeichnis (dem Verzeichnis, das WIN.COM enthält); die Funktion *GetWindowsDirectory* ermittelt den Pfadnamen dieses Verzeichnisses.

3. Im Windows-Systemverzeichnis (dem Verzeichnis, das Systemdateien wie KERNEL.EXE enthält); die Funktion *GetSystemDirectory* ermittelt den Pfadnamen des Verzeichnisses.

4. In den Verzeichnissen, die in der Environment-Variablen PATH aufgeführt sind.

5. In der Liste von Verzeichnissen, die in einem Netz mit dem Befehl *MAP* angegeben wurden.

Wenn der Dateiname einer Anwendung keine andere Namenserweiterung enthält, wird .EXE angenommen.

CmdShow: Beschreibt, wie ein Windows-Anwendungsfenster dargestellt werden soll. Die gültigen Werte für den Parameter finden sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »SW_XXXX«. Bei einer Nicht-Windows-Anwendung legt die .PIF-Datei den Fensterstatus fest.

Rückgabewert Wenn die Funktion erfolgreich war, ist der Rückgabewert größer als 32, andernfalls ist er ein Wert kleiner als 32, der den Fehler näher bestimmt. Mögliche Werte sind:

Wert	Bedeutung
0	Zu wenig freier Speicher.
2	Datei nicht gefunden.
3	Pfad nicht gefunden.
5	Versuch, eine Task dynamisch einzubinden.
6	Bibliothek erfordert getrennte Datensegmente für jede Task.
10	Falsche Windows-Version.
11	Ungültige .EXE-Datei (Keine Windows-.EXE-Datei oder Fehler im .EXE-Darstellungsformat).
12	OS/2-Anwendung.
13	DOS 4.0-Anwendung.
14	Unbekannter .EXE-Typ.
15	Versuch, im Protected Mode (Standardmodus oder erweiterter 386-Modus) eine für frühere Windows-Versionen erstellte .EXE-Datei zu laden.
16	Versuch, eine zweite Instanz einer .EXE-Datei zu laden, die mehrere beschreibbare Datensegmente enthält.
17	Versuch, im Large-Frame-EMS-Modus eine zweite Instanz einer Anwendung zu laden, die mit bestimmten nicht gemeinsam benutzbaren DLLs verbunden ist, die bereits in Gebrauch sind.
18	Versuch, eine Anwendung im Real Mode zu laden, die für Protected Mode geschrieben ist.

Hinweis Die Funktion *LoadModule* bietet eine alternative Methode, ein Programm auszuführen.

WinHelp

Syntax
```
BOOL WinHelp(HWND Wnd, LPSTR HelpFile, WORD Command,
DWORD Data);
```

Beschreibung Diese Funktion ruft die Windows-Hilfeanwendung auf und übergibt optionale Daten, die die Art der von der Anwendung angeforderten Hilfe festlegen. Die Anforderung gibt den Namen und, falls nötig, den Verzeichnispfad der Hilfedatei an, die angezeigt werden soll.

Wnd: Bezeichnet das Fenster, das die Hilfe anfordert.

HelpFile: Zeigt auf einen nullterminierten String, der den Namen der Hilfedatei und, falls nötig, den Verzeichnispfad enthält, die von der Hilfeanwendung angezeigt werden soll.

Command: Bestimmt die Art der angeforderten Hilfe. Er kann einen der in der folgenden Tabelle beschriebenen Werte annehmen:

Wert	Bedeutung
HELP_CONTEXT	Zeigt Hilfe für einen bestimmten Kontext an, der durch den Parameter *Data* bezeichnet wird.
HELP_HELPONHELP	Zeigt Hilfe für die Benutzung der Hilfeanwendung selbst an. Wenn der Parameter *Command* auf *HELP_HELPONHELP* gesetzt wird, ignoriert *WinHelp* die Parameter *HelpFile* und *Data*.
HELP_INDEX	Zeigt den Index der angegebenen Hilfedatei an. Eine Anwendung sollte diesen Wert nur für Hilfedateien mit einem einzigen Index benutzen und sie sollte diesen Wert nicht mit *HELP_SETINDEX* benutzen.
HELP_KEY	Zeigt Hilfe für ein einzelnes Schlüsselwort an, das durch einen String bezeichnet wird, auf den *Data* zeigt.
HELP_MULTIKEY	Zeigt Hilfe für ein Schlüsselwort in einer veränderbaren Schlüsselworttabelle an.
HELP_QUIT	Teilt der Hilfeanwendung mit, daß die angegebene Hilfedatei nicht mehr in Gebrauch ist.
HELP_SETINDEX	Setzt den durch *Data* bestimmten Kontext als aktuellen Index für die durch *HelpFile* angegebene Hilfedatei.

Wert	Bedeutung
	Dieser Index bleibt solange aktuell, bis der Anwender auf eine andere Hilfedatei zugreift. Um sicherzustellen, daß der richtige Index gesetzt bleibt, sollte die Anwendung nach jedem *WinHelp*-Aufruf mit auf *HELP_CONTEXT* gesetztem Parameter *Command* die Funktion *WinHelp* erneut aufrufen, wobei *Command* auf *HELP_SETINDEX* gesetzt sein sollte (und *Data* den zugehörigen Kontextbezeichner angibt). Eine Anwendung sollte diesen Wert nur für Hilfedateien mit mehr als einem Index benutzen und sie sollte diesen Wert nicht mit *HELP_INDEX* verwenden.

Data: Bestimmt den Kontext oder das Schlüsselwort für die angeforderte Hilfe.

- Hat *Command* den Wert *HELP_CONTEXT*, ist *Data* ein vorzeichenloser 32-Bit-Integer, der eine Zahl für den Kontextbezeichner enthält.
- Hat *Command* den Wert *HELP_KEY*, ist *Data* ein Zeiger auf einen nullterminierten String, der ein Schlüsselwort für die Thematik der Hilfe enthält.
- Hat *Command* den Wert *HELP_MULTIKEY*, ist *Data* ein Zeiger auf eine *MULTIKEYHELP*-Datenstruktur.
- In allen anderen Fällen wird *Data* ignoriert und sollte auf 0 gesetzt werden.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Die Anwendung muß *WinHelp* mit auf *HELP_QUIT* gesetztem *Command* aufrufen, bevor sie das Fenster, das die Hilfe anforderte, schließt. Die Hilfeanwendung wird solange nicht beendet, bis alle Anwendungen, die Hilfe angefordert haben, *WinHelp* mit auf *HELP_QUIT* gesetztem *Command* aufgerufen haben.

WriteComm

Syntax
```
int WriteComm(int Cid, LPSTR Buf, int Size);
```

Beschreibung Diese Funktion schreibt die Anzahl der im Parameter *Size* angegebenen Zeichen in die durch *Cid* bezeichnete Schnittstelle. Die Zeichen werden aus dem Datenpuffer entnommen, auf den der Parameter *Buf* zeigt.

Cid: Bestimmt die Schnittstelle, die die Zeichen empfangen soll. Die Funktion *OpenComm* gibt diesen Wert zurück.

Buf: Weist auf den Datenpuffer, der die zu schreibenden Zeichen enthält.

Size: Bestimmt die Zahl der zu schreibenden Zeichen.

Rückgabewert Der Rückgabewert bestimmt die Zahl der gerade geschriebenen Zeichen an. Wenn ein Fehler auftritt, wird der Rückgabewert auf einen Wert kleiner als 0 gesetzt, wobei der Absolutbetrag des Rückgabewerts die Zahl der tatsächlich geschriebenen Zeichen angibt.

Die Fehlerursache kann mit der Funktion *GetCommError* bestimmt werden, die Fehlercode und -status liefert. Eine Übersicht der Fehlercodes findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »CE_XXXX«.

Hinweis *WriteComm* löscht Daten in der Übertragungswarteschlange, wenn dort nicht genügend Platz für zusätzliche Zeichen ist. Anwendungen sollten vor dem Aufruf von *WriteComm* den verfügbaren Platz in der Übertragungswarteschlange mit der Funktion *GetCommError* überprüfen. Außerdem sollten sie die Funktion *OpenComm* benutzen, um die Größe der Übertragungswarteschlange auf einen Wert zu setzen, der nicht kleiner als die Größe des längsten erwarteteten Ausgabestrings ist.

WritePrivateProfileString

Syntax
```
BOOL WritePrivateProfileString(LPSTR ApplicationName,
LPSTR KeyName, LPSTR Str, LPSTR FileName);
```

Beschreibung Diese Funktion kopiert den durch den Parameter *Str* bezeichneten String in die angegebene Initialisierungsdatei. Sie sucht die Datei nach der Schlüsselanweisung ab, die durch den Parameter *KeyName* unter dem Programmkopf der Anwendung benannt ist, der wiederum durch den Parameter *ApplicationName* bestimmt ist. Wird keine Übereinstimmung gefunden, fügt sie dem Benutzerprofil einen neuen Stringeintrag hinzu, der die Bezeichnung der Schlüsselanweisung und den vom Parameter *Str* bestimmten Schlüsselwert enthält. Ist eine übereinstimmende Schlüsselanweisung vorhanden, ersetzt die Funktion diese Schlüsselanweisung durch *Str*.

ApplicationName: Zeigt auf einen nullterminierten String, der den Programmkopf einer Anwendung in der Initialisierungsdatei enthält.

KeyName: Zeigt auf einen nullterminierten String, der eine Schlüsselbezeichnung enthält, die unter dem Programmkopf der Anwendung in der Initialisierungsdatei erscheint.

Str: Zeigt auf den nullterminierten String, der den neuen Wert der Schlüsselanweisung enthält.

FileName: Zeigt auf einen nullterminierten String, der die Initialisierungsdatei bezeichnet. Enthält *FileName* keinen vollständigen Pfadnamen für die Datei, durchsucht *WritePrivateProfileString* das Windows-Verzeichnis nach der Datei. Ist diese nicht vorhanden, wird sie dort erstellt. Enthält *FileName* den vollständigen Pfadnamen einer nicht existierenden Datei, so erstellt *WritePrivateProfileString* keine Datei.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Eine Anwendung sollte eine eigene (anwendungsspezifische) Initialisierungsdatei zum Aufzeichnen von Information, die nur diese Anwendung betrifft, verwenden. Dies erhöht die Ausführungsgeschwindigkeit der Anwendung und von Windows, indem es die Informationsmenge reduziert, die Windows beim Zugriff auf die Initialisierungsdatei lesen muß.

Ein Stringeintrag in der Initialisierungsdatei hat folgende Form:

```
[Anwendungsname]
Schlüsselname = String
.
.
.
```

Eine Anwendung kann *WritePrivateProfileString* auch aufrufen, um Zeilen aus ihrer eigenen Initialisierungsdatei zu löschen. Ist *Str* NULL, löscht die Funktion die ganze durch den Parameter *KeyName* bezeichnete Zeile. Weist *Str* auf einen Nullstring, löscht die Funktion nur den Wert, die Schlüsselbezeichnung bleibt in der Datei. Ist *KeyName* NULL, löscht die Funktion den ganzen durch den Parameter *ApplicationName* bezeichneten Abschnitt, sie löscht jedoch keine Zeilen, die mit dem Kommentarzeichen Strichpunkt (;) beginnen.

WriteProfileString

Syntax
```
BOOL WriteProfileString(LPSTR ApplicationName,
LPSTR KeyName, LPSTR Str);
```

Beschreibung Diese Funktion kopiert den String, auf den der Parameter *Str* zeigt, in die Windows-Initialisierungsdatei WIN.INI. Sie sucht WIN.INI nach der Schlüsselanweisung ab, die durch den Parameter *KeyName* unter dem Kopf der Anwendung benannt wird. Das Anwendungsprogramm selbst wird durch den Parameter *ApplicationName* bezeichnet.

Wird keine Übereinstimmung gefunden, fügt *WriteProfile-*

String dem Benutzerprofil einen neuen Stringeintrag hinzu, der die Bezeichnung der Schlüsselanweisung und des Schlüsselwertes enthält. Ist eine entsprechende Schlüsselanweisung vorhanden, ersetzt die Funktion den Wert der Schlüsselanweisung durch *Str*.

ApplicationName: Zeigt auf einen nullterminierten String, der den Kopf eines Anwendungsprogramms in WIN.INI enthält.

KeyName: Zeigt auf einen nullterminierten String, der die Schlüsselbezeichnung enthält, die unter dem Kopf eines Anwendungsprogrammes in WIN.INI erscheint.

Str: Zeigt auf den nullterminierten String, der den neuen Wert der Schlüsselanweisung enthält.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn die Funktion erfolgreich war, andernfalls ist er Null.

Hinweis Ein Stringeintrag in WIN.INI hat folgende Form:

```
[Anwendungsname]
Schlüsselname = String
. . .
```

Eine Anwendung kann *WriteProfileString* auch aufrufen, um Zeilen aus WIN.INI zu löschen. Ist *Str* NULL, löscht die Funktion die ganze durch den Parameter *KeyName* bezeichnete Zeile. Zeigt *Str* auf einen Nullstring, löscht die Funktion nur den Wert, die Schlüsselbezeichnung bleibt in der Datei. Ist *KeyName* NULL, löscht die Funktion den gesamten, durch den Parameter *ApplikationName* bezeichneten Abschnitt, sie löscht jedoch keine Zeilen, die mit einem Strichpunkt (;) als Kommentarzeichen beginnen.

wsprintf

Syntax
```
int wsprintf(LPSTR Output, LPSTR Format
[[, Argument]]...)
```

Beschreibung Diese Funktion formatiert und speichert eine Folge von Zeichen und Werten in einem Puffer.

Die Elemente, auf die die Argumentliste zeigt, werden gemäß der zugehörigen Formatvorgabe konvertiert und ausgegeben. Die Funktion fügt eine Null an das Ende der geschriebenen Zeichen an. Der Rückgabewert berücksichtigt das abschließende Nullzeichen bei der Zeichenzählung nicht.

Output: Zeigt auf einen nullterminierten String, der die formatierte Ausgabe aufnehmen soll.

Format: Zeigt auf einen nullterminierten String, der die Formatvorgabe enthält. Zusätzlich zu den gewöhnlichen ASCII-Zei-

chen erscheint in diesem String eine Formatvorgabe für jedes Argument.

Argument: Ein oder mehrere Argumente. Die Anzahl, der Typ und die Interpretation des Arguments hängen von den entsprechenden Formatangaben im Parameter *Format* ab.

Rückgabewert

Der Rückgabewert ist die Anzahl der in *Output* gespeicherten Zeichen, die abschließende Null wird dabei nicht mitgezählt. Wenn ein Fehler auftritt, gibt die Funktion einen Wert zurück, der kleiner als die Länge von *Format* ist.

Hinweis

Der Formatstring enthält Formatspezifikationen, die das Ausgabeformat der folgenden Argumente bestimmen. Formatspezifikationen beginnen immer mit einem Prozentzeichen (%). Folgt auf das Prozentzeichen ein weiteres Zeichen ohne besondere Bedeutung, wird dieses Zeichen ohne weitere Formatierung ausgegeben. Die Zeichenfolge %% führt also zur Ausgabe eines einzelnen Prozentzeichens.

Der Formatstring wird von links nach rechts gelesen. Das erste Argument, das auf eine Formatspezifikation folgt, wird entsprechend dieser Anweisung formatiert und ausgegeben. Die nächste Formatspezifikation bezieht sich auf das zweite Argument, usw. Existieren mehr Argumente als Formatspezifikationen, werden die überzähligen Argumente ignoriert. Sind dagegen die Formatspezifikationen in der »Überzahl«, kann es zu unvorhersehbaren Resultaten kommen.

Eine Formatspezifikation hat folgende Form:

```
% [-] [#] [0] [Breite] [.Präzision] [Typ]
```

Jedes Feld der Formatspezifikation ist ein Zeichen oder eine Ziffer, die jeweils bestimmte Formatoptionen aktivieren. Diese optionalen Felder werden in folgender Tabelle beschrieben.

Feld	Beschreibung
–	Die Ausgabe wird rechts von Nullen oder Leerstellen ergänzt, um die Feldbreite auszufüllen. Die Ausgabe wird linksbündig ausgerichtet. Wird dieses Feld nicht verwendet, werden die Zeichen links angefügt und die Ausgabe rechtsbündig formatiert.
#	Präfix für Hexademimalwerte (0x)
0	Die Feldbreite wird mit Nullen aufgefüllt. Wird diese Option nicht verwendet, werden Leerzeichen als Füllzeichen verwendet.
Breite	Die Mindestbreite der Ausgabe. Der Wert muß ein positiver Integer sein. Wird diese Option nicht verwendet oder ist die Anzahl der Zeichen größer als angegeben, werden trotzdem alle Zeichen unter Berücksichtigung des Feldes *Präzision* ausgegeben.

Feld	Beschreibung
Präzision	Gibt mindestens die angegebene Anzahl von Ziffern aus. Ist die Anzahl der Ziffern im Argument kleiner, wird die Ausgabe links mit Nullen aufgefüllt. Werte, die größer sind, werden nicht abgeschnitten. Hat *Präzision* den Wert 0, wird das Feld ignoriert, hat es nur einen Punkt (.) zum Inhalt, wird es auf 1 gesetzt. Strings werden mit der angegebenen maximalen Zeichenzahl ausgegeben.
Typ	Gibt das entsprechende Argument als Zeichen, Zahl oder String aus. *Typ* kann mit folgenden Zeichenfolgen benutzt werden:
	s — Fügt ein String-Argument ein, auf das ein LPSTR zeigen muß.
	c — Fügt ein einzelnes Zeichen ein.
	d, i — Fügt ein dezimales *signed int*-Argument ein.
	ld, li — Fügt ein dezimales *signed longint*-Argument ein.
	u — Fügt ein dezimales *unsigned int*-Argument ein.
	lu — Fügt ein dezimales *unsigned longint*-Argument ein.
	x, x — Fügt ein hexadezimales *unsigned int*-Argument in Klein- oder Großschreibung ein.
	lx, lX — Fügt ein hexadezimales *unsigned longint*-Argument in Klein- oder Großschreibung ein.

Im Gegensatz zu allen anderen Windows-Funktionen verwendet *wsprintf* die C-Aufrufkonvention (*cdecl*). Es ist also Aufgabe des Aufrufers, die Argumente vom Stack zu nehmen und in umgekehrter Reihenfolge wieder abzulegen. Der Parameter *Output* muß also an der untersten Adresse liegen. In C-Modulen übernimmt der C-Compiler diese Aufgabe.

wvsprintf

Syntax

```
int wvsprintf(LPSTR Output, LPSTR Format, LPSTR
ArgList);
```

Beschreibung

Diese Funktion formatiert und speichert eine Folge von Zeichen in einem Puffer.

Die Elemente, auf die die Argumentliste zeigt, werden gemäß der zugehörigen Formatvorgabe konvertiert und ausgegeben. Die Funktion fügt eine 0 an das Ende der geschriebenen Zeichen. Der Rückgabewert berücksichtigt das abschließende Nullzeichen bei der Zeichenzählung nicht.

Output: Zeigt auf einen nullterminierten String, der die formatierte Ausgabe aufnehmen soll.

Format: Zeigt auf einen nullterminierten String, der die Formatvorgabe enthält. Zusätzlich zu den gewöhnlichen ASCII-Zei-

chen erscheint in diesem String eine Formatvorgabe für jedes Argument.

ArgList: Zeigt auf ein Argument-Array, in dem jedes WORD ein Argument für die Formatvorgabe bezeichnet. Die Anzahl, der Typ und die Interpretation des Arguments hängen von den entsprechenden Zeichensequenzen der Formatangaben im Parameter *Format* ab.

- Jedes Zeichen oder jede ein WORD umfassende Integerzahl (%c, %d, %x, %i) erfordert ein WORD in *ArgList*.
- LONG-Werte (%ld, %li, %lx) erfordern zwei WORD, das niederwertige gefolgt vom höherwertigen WORD.
- Ein String (%s) benötigt zwei WORD, den Offset gefolgt vom Segment (die zusammen einen **far**-Zeiger ergeben).

Rückgabewert Der Rückgabewert ist die Anzahl der in *Output* gespeicherten Zeichen, die abschließende 0 wird dabei nicht mitgezählt. Wenn ein Fehler auftritt, gibt die Funktion einen Wert zurück, der kleiner als die Länge von *Format* ist.

Yield

Syntax `void Yield(void);`

Beschreibung Diese Funktion unterbricht die aktuelle Task und startet beliebige wartende Tasks.
Yield hat keinen Parameter.

Rückgabewert Keiner.

Anmerkungen Anwendungen, die mit Fenstern arbeiten, sollten besser eine *DispatchMessage*-, *PeekMessage*- oder *TranslateMessage*-Schleife einsetzen, als die Funktion *Yield* zu verwenden. Die *PeekMessage*-Schleife führt die Synchronisation von Botschaften genauer durch und unterbricht zur richtigen Zeit.

2. Windows-Botschaften

In diesem Kapitel werden alle Windows-Botschaften alphabetisch aufgeführt. Zu jeder Botschaft findet sich eine Beschreibung ihrer Verwendungsmöglichkeit, ihrer Parameter sowie des Rückgabewertes, der von Windows erwartet oder zurückgeliefert wird. Die Parameter *wParam* und *lParam* gehören zu jeder Botschaft. Eine genaue Beschreibung der *MSG*-Struktur mit den Feldern *wParam* und *lParam* findet sich in Kapitel 3 »Windows-Typen« im Abschnitt »MSG-Typ«.

Die Parameter *lParam* und *wParam* sind Werte vom Typ *LONG* bzw. *WORD*. Um diesen Werten die einzelnen Informationsbestandteile zu entnehmen, können die Makros *HIBYTE*, *LOBYTE*, *HIWORD* und *LOWORD* verwendet werden. Sie ermöglichen den Zugriff auf die höherwertigen und niederwertigen Bytes bzw. WORD-Werte, die in den Parametern *wParam* und *lParam* übergeben werden.

Außer diesen beiden Parametern werden auch die Bezeichner der jeweiligen Botschaft und des empfangenden Fensters übergeben.

BM_GETCHECK

Beschreibung	Diese Botschaft ermittelt, ob ein Auswahlfeld oder Schaltfeld markiert ist.
Parameter	*wParam:* Nicht verwendet.
	lParam: Nicht verwendet.
Rückgabewert	Der Rückgabewert ist ungleich 0, wenn das Auswahlfeld oder das Schaltfeld markiert ist, andernfalls ist er 0. Die Botschaft *BM_GETCHECK* gibt für einen Schalter immer 0 zurück.

BM_GETSTATE

Beschreibung	Diese Botschaft bestimmt den Status eines Schalters, wenn der Anwender eine Maustaste oder die Leertaste drückt.
Parameter	*wParam:* Nicht verwendet.
	lParam: Nicht verwendet.
Rückgabewert	Die Botschaft *BM_GETSTATE* gibt einen Wert ungleich 0

zurück, wenn eine der folgenden Bedingungen erfüllt ist, andernfalls ist der Rückgabewert 0:

- Ein Schalter ist hervorgehoben.
- Der Anwender drückt eine Maustaste oder die Leertaste, wenn ein Schalter den Fokus hat.
- Der Anwender drückt eine Maustaste, während der Mauszeiger auf einem Schalter steht.

BM_SETCHECK

Beschreibung	Diese Botschaft markiert ein Schaltfeld oder ein Auswahlfeld oder löscht die Auswahlmarkierung von einem Schaltfeld oder Auswahlfeld.
Parameter	*wParam:* Bestimmt, ob eine Auswahlmarkierung für einen Schalter oder ein Auswahlfeld gesetzt oder gelöscht werden soll. Ist der Parameter *wParam* ungleich 0, wird eine Auswahlmarkierung gesetzt, ist er 0, wird sie entfernt. Bei dreistufigen Schaltern wird eine Auswahlmarkierung neben den Schalter gesetzt, wenn *wParam* 1 ist. Ist *wParam* 2, wird der Schalter in den nicht aktivierbaren Zustand überführt. Ist *wParam* 0, wird der Schalter in den normalen Zustand zurückgesetzt (keine Markierung, aber aktivierbar). *lParam*: Nicht verwendet.
Rückgabewert	Nicht verwendet.
Hinweis	Die Botschaft *BM_SETCHECK* hat keinen Einfluß auf Aktionsschalter.

BM_SETSTATE

Beschreibung	Diese Botschaft ändert den Status eines Schalters oder eines Auswahlfeldes.
Parameter	*wParam:* Bestimmt die Art der Hervorhebung des Schalters. Ist der Parameter *wParam* ungleich 0, wird der Schalter hervorgehoben (das Innere wird invertiert dargestellt). Ist *wParam* 0, wird der Schalter in seinem regulären Status gezeichnet. *lParam*: Nicht verwendet.
Rückgabewert	Nicht verwendet.
Hinweis	Aktionsschalter können nicht hervorgehoben werden.

BM_SETSTYLE

Beschreibung Diese Botschaft ändert den Stil eines Schalters. Weicht der Typ, der im Parameter *wParam* enthalten ist, vom vorgefundenen Stil ab, wird der Schalter in der neuen Form gezeichnet.

Parameter *wParam:* Gibt den Wert für den Stil an. Eine Beschreibung aller Schalterstile findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »BS_XXXX«.

lParam: Bestimmt, ob die Schalter neu gezeichnet werden sollen. Ist *lParam* 0, werden die Schalter nicht neu gezeichnet. Ist *lParam* ungleich 0, werden sie neu gezeichnet.

Rückgabewert Nicht verwendet.

BN_-Benachrichtigungen

Hinweis Die BN_-Benachrichtigungen sind keine Windows-Botschaften im eigentlichen Sinn. Ihre Beschreibung finden Sie in Kapitel 4 »Windows-Stile und -Konstanten«.

CB_ADDSTRING

Beschreibung Diese Botschaft fügt einen String zur Liste eines Kombinationsfensters hinzu. Wenn die Liste nicht sortiert ist, wird der String am Ende der Liste angefügt, ansonsten wird der String nach erneutem Sortieren in die Liste eingefügt.

Die Botschaft löscht jede vorhandene Listenauswahl.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf den nullterminierten String, der hinzugefügt werden soll. Wurde das Kombinationsfenster ohne Verwendung von *CBS_HASSTRINGS* durch die Anwendung erstellt, ist der Parameter *lParam* ein von der Anwendung übergebener 32-Bit-Wert, der vom Kombinationsfenster anstelle des Zeigers auf den String gespeichert wird.

Rückgabewert Der Rückgabewert ist der Index zu String und Liste. Wenn ein Fehler auftritt, wird *CB_ERR*, wenn zum Speichern des neuen Strings nicht genügend Speicher zur Verfügung steht, *CB_ERRSPACE* zurückgegeben.

Hinweis Wird das Kombinationsfenster einer Anwendung mit *CBS_SORT*, aber ohne *CBS_OWNERDRAWFIXED* bzw. *CBS_OWNERDRAWVARIABLE* und *CBS_HASSTRINGS* erstellt , wird die Botschaft *WM_COMPAREITEM* ein- oder mehrmals an den Besitzer des Kombinationsfensters gesendet, so daß der neue Eintrag richtig in der Liste eingeordnet werden kann.

CB_DELETESTRING

Beschreibung Diese Botschaft löscht einen String aus der Liste eines Kombinationsfensters.

Parameter *wParam:* Enthält den Index des Strings, der gelöscht werden soll.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Anzahl der Strings, die in der Liste verbleiben. Der Rückgabewert ist *CB_ERR*, wenn *wParam* keinen gültigen Index bezeichnet.

Hinweis Wenn das Kombinationsfenster von der Anwendung gezeichnet ist (entweder *CBS_OWNERDRAWFIXED* oder *CBS_OWNERDRAWVARIABLE*), aber ohne *CBS_HASSTRINGS* erstellt wurde, wird eine Botschaft *WM_DELETEITEM* an den Besitzer des Kombinationsfensters gesendet, so daß die Anwendung zusätzliche, dem Eintrag durch den Parameter *lParam* der Botschaften *CB_ADDSTRING* oder *CB_INSERTSTRING* zugeordnete Daten freigeben kann.

CB_DIR

Beschreibung Diese Botschaft fügt eine Liste der Dateien aus dem aktuellen Verzeichnis zur Liste hinzu. Es werden nur Dateien mit den von *wParam* vorgegebenen Attributen und entsprechend der vom Parameter *lParam* bezeichneten Dateispezifikation hinzugefügt.

Parameter *wParam:* Enthält einen DOS-Attributwert. Eine Liste der DOS-Attribute findet sich in der Beschreibung der Funktion *DlgDirList*.

lParam: Zeigt auf einen String, der die Dateinamen spezifiziert. Der nullterminierte String kann Platzhalterzeichen enthalten (zum Beispiel *.*).

Rückgabewert Der Rückgabewert ist der Index des letzten Eintrags in der Liste. Der Rückgabewert ist *CB_ERR*, wenn ein Fehler auftritt und *CB_ERRSPACE*, wenn zum Speichern des neuen Strings nicht genügend Speicherplatz vorhanden ist.

Hinweis Der Index des ersten Listeneintrags ist 0. Deshalb ist der Rückgabewert der Botschaft *CB_DIR* um 1 kleiner als der Rückgabewert der Botschaft *CB_GETCOUNT*.

CB_FINDSTRING

Beschreibung Diese Botschaft findet den ersten String in der Liste eines Kombinationsfensters, der mit dem angegebenen Präfixstring übereinstimmt.

Parameter *wParam:* Enthält den Index zu dem Eintrag vor dem ersten zu überprüfenden Eintrag. Wenn die Suche das Ende einer Liste erreicht, wird sie vom Listenbeginn an bis zu dem von *wParam* bezeichneten Element fortgesetzt. Ist der Parameter *wParam* -1, wird die ganze Liste von Beginn an durchsucht.

lParam: Zeigt auf den Präfixstring. Der String muß nullterminiert sein.

Rückgabewert Der Rückgabewert ist der Index des ersten übereinstimmenden Eintrags oder *CB_ERR*, wenn die Suche nicht erfolgreich war.

Hinweis Wenn das Kombinationsfenster von der Anwendung gezeichnet ist, aber ohne *CBS_HASSTRINGS* erstellt wurde, gibt diese Botschaft den Index des Eintrags zurück, dessen zugehöriger 32-Bit-Wert (übergeben als Parameter *lParam* der Botschaften *CB_ADDSTRING* oder *CB_INSERTSTRING*) mit dem Wert übereinstimmt, der als Parameter *lParam* von *CB_FINDSTRING* übergeben wurde.

CB_GETCOUNT

Beschreibung Diese Botschaft gibt die Anzahl der Einträge in der Liste eines Kombinationsfensters zurück.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Anzahl der Einträge in der Liste eines Kombinationsfensters.

CB_GETCURSEL

Beschreibung Diese Botschaft gibt den Index des aktuell ausgewählten Eintrags in der Liste eines Kombinationsfensters zurück, falls ein solcher existiert.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist der Index des aktuell ausgewählten Eintrags. Er ist *CB_ERR*, wenn kein Eintrag ausgewählt ist.

CB_GETEDITSEL

Beschreibung Diese Botschaft gibt die Start- und Endpositionen des ausgewählten Texts im Editierfeld eines Kombinationsfensters zurück.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist ein LONG-Wert, der die Startposition im niederwertigen und die Endposition im höherwertigen WORD enthält. Wenn die Botschaft an ein Kombinationsfenster ohne Editierfeld gesendet wurde, ist der Rückgabewert *CB_ERR*.

CB_GETITEMDATA

Beschreibung Diese Botschaft liefert den von der Anwendung bereitgestellten 32-Bit-Wert, der dem angegebenen Kombinationsfenstereintrag zugeordnet ist.

Wenn sich der Eintrag in einem von der Anwendung gezeichneten Kombinationsfenster befindet, das ohne *CBS_HASSTRINGS* erstellt wurde, war dieser 32-Bit-Wert im Parameter *lParam* der Botschaften *CB_ADDSTRING* oder *CB_INSERTSTRING* enthalten, die den Eintrag zum Kombinationsfenster hinzufügten.

Andernfalls ist es der Wert des Parameters *lParam* einer Botschaft *CB_SETITEMDATA*.

Parameter *wParam:* Enthält den Index eines Eintrags.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist der 32-Bit-Wert, der dem Eintrag zugeordnet ist oder *CB_ERR*, wenn ein Fehler auftritt.

CB_GETLBTEXT

Beschreibung Diese Botschaft kopiert einen String aus der Liste eines Kombinationsfensters in einen Puffer.

Parameter *wParam:* Enthält den Index des Strings, der kopiert werden soll.

lParam: Zeigt auf einen Puffer, der den String aufnehmen soll. Der Puffer muß groß genug sein, um den String und ein abschließendes Nullzeichen aufzunehmen.

Rückgabewert Der Rückgabewert ist die Länge des Strings in Bytes, ohne Berücksichtigung des abschließenden Nullzeichens. Bezeichnet *wParam* keinen gültigen Index, ist der Rückgabewert *CB_ERR*.

Hinweis Wurde das Kombinationsfenster mit *CBS_OWNERDRAWFIXED* oder *CBS_OWNERDRAWVARIABLE*, jedoch ohne *CBS_HASSTRINGS* erstellt, so empfängt der Puffer, auf den der Parameter *lParam* zeigt, den 32-Bit-Wert, der dem Eintrag durch den Parameter *lParam* der Botschaften *CB_ADDSTRING* oder *CB_INSERTSTRING* zugeordnet wurde.

CB_GETLBTEXTLEN

Beschreibung Diese Botschaft gibt die Länge eines Strings in der Liste eines Kombinationsfensters in Bytes zurück.

Parameter *wParam:* Enthält den Index des Strings.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Länge des Strings in Bytes, ohne Berücksichtigung des abschließenden Nullzeichens. Wenn der Parameter *wParam* keinen gültigen Index enthält, ist der Rückgabewert *CB_ERR*.

CB_INSERTSTRING

Beschreibung Diese Botschaft fügt einen String in die Liste eines Kombinationsfensters ein. Die Liste wird nicht sortiert.

Parameter *wParam:* Enthält den Index der Position, die den String aufnehmen soll. Wenn der Parameter *wParam* -1 ist, wird der String am Ende der Liste angefügt.

lParam: Zeigt auf den nullterminierten String, der eingefügt werden soll. Wurde das Kombinationsfenster ohne den Stil *CBS_HASSTRINGS* erstellt, ist *lParam* ein von der Anwendung bereitgestellter 32-Bit-Wert, der vom Kombinationsfenster anstelle des Zeigers auf den String gespeichert wird.

Rückgabewert Der Rückgabewert ist der Index der Position, an der der String eingefügt wurde. Wenn ein Fehler auftritt, ist der Rückgabewert *CB_ERR*. Steht zum Speichern des neuen Strings nicht genügend Speicherplatz zur Verfügung, ist er *CB_ERRSPACE*.

CB_LIMITTEXT

Beschreibung Diese Botschaft begrenzt die Länge des Textes, den der Anwender in das Editierfeld eines Kombinationsfensters eingeben kann, in Bytes.

Parameter *wParam:* Bestimmt die Anzahl von Bytes, die der Anwender höchstens eingeben kann. Is *wParam* 0, gibt es keine Beschränkung.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn die Botschaft erfolgreich war, andernfalls ist er 0.

Wird diese Botschaft an ein Kombinationsfenster ohne Editierfeld gesendet, ist der Rückgabewert *CB_ERR*.

CB_RESETCONTENT

Beschreibung Diese Botschaft löscht alle Strings aus der Liste eines Kombinationsfensters und gibt den für diese Strings reservierten Speicherplatz frei.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet

Hinweis Wenn das Kombinationsfenster ohne *CBS_HASSTRINGS*, jedoch mit *CBS_OWNERDRAWFIXED* oder *CBS_OWNERDRAWVARIABLE* erstellt wurde, erhält der Besitzer des Kombinationsfensters eine *WM_DELETEITEM*-Botschaft für jeden Eintrag im Kombinationsfenster.

CB_SELECTSTRING

Beschreibung Diese Botschaft wählt den ersten String in der Liste eines Kombinationsfensters aus, der mit dem angegebenen Präfix übereinstimmt. Der Text im Editierfeld des Kombinationsfensters wird verändert, um die neue Auswahl widerzuspiegeln.

Parameter *wParam:* Enthält den Index zu dem Eintrag vor dem ersten Eintrag, ab dem gesucht werden soll. Wenn die Suche das Ende der Liste erreicht, wird sie vom Listenbeginn an bis zu dem von *wParam* bezeichneten Eintrag fortgeführt. Wenn *wParam* -1 ist, wird die ganze Liste von Beginn an durchsucht.

lParam: Zeigt auf den Präfixstring. Dieser muß ein abschliessendes Nullzeichen enthalten.

Rückgabewert Der Rückgabewert ist der Index des ersten übereinstimmenden Eintrags. Wenn die Suche nicht erfolgreich war, ist der Rückgabewert *CB_ERR* und die aktuelle Auswahl wird nicht verändert.

Hinweis Ein String wird nur ausgewählt, wenn seine Anfangsbuchstaben mit den Zeichen im Präfixstring übereinstimmen.

Wenn das Kombinationsfenster ohne *CBS_HASSTRINGS* erstellt wurde, gibt diese Botschaft den Index zu dem Eintrag zurück, dessen zugehöriger 32-Bit-Wert (übergeben als Parameter *lParam* der Botschaften *CB_ADDSTRING* oder *CB_-*

INSERTSTRING) mit dem Wert übereinstimmt, der als Parameter *lParam* von *CB_FINDSTRING* übergeben wurde.

CB_SETCURSEL

Beschreibung Diese Botschaft wählt einen String aus der Liste eines Kombinationsfensters aus und führt Bildlauf in den sichtbaren Bereich durch, wenn die Liste angezeigt ist. Außerdem wird der Text im Editierfeld oder im statischen Textfeld des Kombinationsfensters geändert, um die neue Auswahl widerzuspiegeln. Wenn der neue String ausgewählt ist, entfernt die Liste die Hervorhebung des vorher ausgewählten String.

Parameter *wParam:* Enthält den Index des Strings, der ausgewählt werden soll. Wenn *wParam* -1 ist, wird der Liste keine Auswahl zugewiesen.

lParam: Nicht verwendet.

Rückgabewert Wenn der von *wParam* bezeichnete Index ungültig ist, ist der Rückgabewert *CB_ERR* und die aktuelle Auswahl wird nicht verändert.

CB_SETEDITSEL

Beschreibung Diese Botschaft wählt alle Zeichen im Editierfeld eines Kombinationsfensters aus, die innerhalb der Start- und Endpositionen liegen, die durch den Parameter *lParam* bestimmt werden.

Parameter *wParam:* Nicht verwendet.

lParam: Der niederwertige Integerwert von *lParam* ist der Zeichenindex der Startposition. Der höherwertige Integerwert von *lParam* ist der Zeichenindex der Endposition.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn die Botschaft erfolgreich war, andernfalls ist er 0.

Wenn die Botschaft an ein Kombinationsfenster ohne Editierfeld gesendet wird, ist der Rückgabewert *CB_ERR*.

CB_SETITEMDATA

Beschreibung Diese Botschaft setzt den 32-Bit-Wert, der dem angegebenen Eintrag in einem Kombinationsfenster zugeordnet ist.

Wenn sich der Eintrag in einem von der Anwendung gezeichneten Kombinationsfenster befindet, das ohne *CBS_HASSTRINGS* erstellt wurde, ersetzt diese Botschaft den 32-Bit-Wert, der im Parameter *lParam* der Botschaften *CB_ADDSTRING* oder *CB_INSERTSTRING* enthalten war, die den Eintrag zum Kombinationsfenster hinzugefügt haben.

Parameter *wParam:* Enthält den Index des Eintrags.

lParam: Enthält den neuen 32-Bit-Wert, der dem Eintrag zugeordnet werden soll.

Rückgabewert Der Rückgabewert ist *CB_ERR*, wenn ein Fehler auftritt.

CB_SHOWDROPDOWN

Beschreibung Diese Botschaft zeigt die Drop-Down-Liste eines Kombinationsfenster, das mit dem Stil *CBS_DROPDOWN* oder *CBS_DROPDOWNLIST* erstellt wurde bzw. blendet sie aus.

Parameter *wParam:* Ist dieser Parameter ungleich 0, wird die Liste dargestellt, wenn sie nicht sichtbar ist. Ist er 0, wird die Liste ausgeblendet, wenn sie sichtbar ist.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

CBN_-Benachrichtigungen

Hinweis Die CBN_-Benachrichtigungen sind keine Windows-Botschaften im eigentlichen Sinn. Ihre Beschreibung finden Sie in Kapitel 4 »Windows-Stile und -Konstanten«.

DM_GETDEFID

Beschreibung Diese Botschaft liefert die ID des als Vorgabewert eingestellten Aktionsschalters für ein Dialogfenster.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist ein 32-Bit-Wert. Das höherwertige WORD enthält *DC_HASDEFID*, falls ein Vorgabeschalter existiert, andernfalls ist es 0. Das niederwertige WORD enthält die ID des Vorgabeschalters, wenn das höherwertige WORD *DC_HASDEFID* enthält; andernfalls ist es 0.

DM_SETDEFID

Beschreibung Diese Botschaft wird von einer Anwendung benutzt, um die ID für den als Vorgabewert vorgesehenen Aktionsschalter eines Dialogfensters zu ändern.

Parameter *wParam:* Enthält die ID des neuen Vorgabeschalters.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

EM_CANUNDO

Beschreibung Diese Botschaft bestimmt, ob ein Editierfeld korrekt auf eine *EM_UNDO*-Botschaft reagieren kann.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn das Editierfeld auf die Botschaft *EM_CANUNDO* korrekt reagieren kann, andernfalls ist er Null.

EM_EMPTYUNDOBUFFER

Beschreibung Diese Botschaft weist ein Editierfeld an, seinen Widerrufen-Puffer zu löschen. Dies nimmt dem Editierfeld die Fähigkeit, den letzten Editiervorgang rückgängig zu machen.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Der Puffer zum Widerrufen wird automatisch geleert, wenn das Editierfeld eine Botschaft *WM_SETTEXT* oder *EM_SETHANDLE* empfängt.

EM_FMTLINES

Beschreibung Diese Botschaft weist ein Mehrzeilen-Editierfeld an, das Zeilenendezeichen aus umgebrochenen Zeilen zu entfernen oder es ihnen hinzuzufügen.

Parameter *wParam:* Bestimmt die Verwendung von Zeilenendezeichen. Ist *wParam* ungleich 0, werden die Zeichen *CR CR LF* (0D 0D 0A hexadezimal) an das Ende der umbrochenen Zeilen gesetzt, ist er 0, werden die Zeilenendezeichen aus dem Text gelöscht.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn der Text verändert wurde, andernfalls ist er 0.

Hinweis Zeilen, die mit einem Hard-Return enden (einem Wagenrücklaufzeichen, das vom Anwender eingegeben wurde), erhalten die Zeichen *CR LF* am Zeilenende. Diese Zeilen werden von der Botschaft *EM_FMTLINES* nicht berücksichtigt.

Der Umfang des Textes ändert sich, wenn diese Botschaft verarbeitet wird.

EM_GETHANDLE

Beschreibung Diese Botschaft gibt das Daten-Handle des Puffers zurück, der den Inhalt des Editierfeldes enthält. Das Handle ist immer ein lokales Handle auf eine Position im Datensegment der Anwendung.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist das Daten-Handle des Puffers, der den Inhalt des Editierfeldes enthält.

Hinweis Eine Anwendung kann diese Botschaft nur an Editierfelder übergeben, deren Dialogfenster mit gesetztem Flag *DS_LOCALEDIT* erzeugt wurde.

EM_GETLINE

Beschreibung Diese Botschaft kopiert eine Zeile aus einem Editierfeld.

Parameter *wParam:* Bestimmt die Zeilennummer der Zeile im Steuerelement, wobei die Zeilennummer der ersten Zeile 0 ist.

lParam: Zeigt auf den Puffer, in dem die Zeile gespeichert werden soll.

Das erste WORD des Puffers gibt die maximale Anzahl von Bytes an, die in den Puffer kopiert werden sollen. Die kopierte Zeile endet nicht mit einem Nullzeichen.

Rückgabewert Der Rückgabewert ist die Anzahl von Bytes, die tatsächlich kopiert wurden.

Diese Botschaft wird von Einzeileneditierfeldern nicht bearbeitet.

EM_GETLINECOUNT

Beschreibung Diese Botschaft gibt die Anzahl der Textzeilen eines Editierfelds zurück.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Anzahl der Textzeilen im Steuerelement.

Hinweis Diese Botschaft wird von Einzeileneditierfeldern nicht verarbeitet.

EM_GETMODIFY

Beschreibung Diese Botschaft gibt den aktuellen Wert des Veränderungsflags für ein gegebenes Editierfeld zurück.

Das Flag wird vom Editierfeld auf einen Wert ungleich 0 gesetzt, wenn der Anwender Text innerhalb des Editierfeldes eingibt oder bearbeitet.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist der Wert des aktuellen Veränderungsflags für ein vorgegebenes Editierfeld.

EM_GETRECT

Beschreibung Diese Botschaft liefert das formatierende Rechteck eines Editierfeldes.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf eine *RECT*-Datenstruktur, die durch diese Botschaft gefüllt wird.

Rückgabewert Nicht verwendet.

EM_GETSEL

Beschreibung Diese Botschaft gibt die Indizes der Start- und Endzeichen der aktuellen Auswahl in einem Editierfeld zurück.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert enthält die Startposition im niederwertigen WORD. Im höherwertigen WORD erhält er die Position des ersten nicht ausgewählten Zeichens nach dem Ende der Auswahl.

EM_LIMITTEXT

Beschreibung Diese Botschaft begrenzt die Anzahl der Zeichen, die der Anwender in ein Editierfeld eingeben kann.

Parameter *wParam:* Bestimmt die maximale Anzahl von Bytes, die eingegeben werden können. Wenn der Anwender versucht, mehr

Zeichen einzugeben, wird ein Warnton ausgegeben und die Zeichen werden nicht angenommen. Ist der Parameter *wParam* 0, wird die Textlänge nur durch den verfügbaren Speicherplatz begrenzt.

lParam: Nicht verwendet.

Rückgabewert Tritt ein Fehler auf, so ist der Rückgabewert 0, andernfalls ungleich 0.

Hinweis Die Botschaft *EM_LIMITTEXT* betrifft keinen Text, der von der Botschaf *WM_SETTEXT* gesetzt wurde und keinen Puffer, der von der Botschaft *EM_SETHANDLE* eingerichtet wurde.

EM_LINEFROMCHAR

Beschreibung Diese Botschaft gibt die Zeilennummer der Zeile zurück, die das Zeichen enthält, dessen Index vom Parameter *wParam* vorgegeben ist.

Parameter *wParam:* Enthält den Indexwert für das gewünschte Zeichen im Text des Editierfeldes (diese Indexwerte beginnen mit 0) oder -1.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist eine Zeilennummer. Ist *wParam* -1, wird die Nummer der Zeile zurückgegeben, die das erste Zeichen der Auswahl enthält. Andernfalls enthält *wParam* den Index (oder die Position) des gewünschten Zeichens im Text des Editierfeldes und es wird die Nummer der Zeile, die das Zeichen enthält, zurückgegeben.

EM_LINEINDEX

Beschreibung Diese Botschaft gibt die Anzahl von Zeichenpositionen zurück, die vor dem ersten Zeichen einer angegebenen Zeile auftreten.

Parameter *wParam:* Bezeichnet die gewünschte Zeilennummer, wobei die Zeilennummer der ersten Zeile 0 ist. Ist der Parameter *wParam* -1, wird die aktuelle Zeilennummer (die Zeile, die das Caret enthält) verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Anzahl von Zeichenpositionen, die dem ersten Zeichen in der vorgegebenen Zeile vorangehen.

Hinweis Die Botschaft wird von Einzeileneditierfeldern ignoriert.

EM_LINELENGTH

Beschreibung Diese Botschaft gibt die Länge einer Zeile im Textpuffer eines Editierfeldes in Bytes zurück.

Parameter *wParam:* Bezeichnet den Zeichenindex eines Zeichens in der gewünschten Zeile. Der Index des ersten Zeichens ist 0.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Länge der Zeile, die das Zeichen mit dem angegebenen Index enthält, abzüglich der Länge eines eventuell markierten Textes in dieser Zeile. Umfaßt die aktuelle Markierung mehrere Zeilen, wird die gesamte Länge dieser Zeilen abzüglich der Länge des markierten Textes zurückgegeben.

Hinweis Einen Zeichenindex zu einer gegebenen Zeilennummer, wie er für die Botschaft *EM_LINELENGTH* benötigt wird, erhält man mittels der Botschaft *EM_LINEINDEX*.

EM_LINESCROLL

Beschreibung Diese Botschaft rollt den Inhalt eines Editierfeldes.

Parameter *wParam:* Nicht verwendet.

lParam: Der niederwertige Integerwert von *lparam* ist die Anzahl der Zeilen, um die vertikaler Bildlauf durchgeführt werden soll. Der höherwertige Integerwert von *lparam* ist die Anzahl der Spalten, um die horizontaler Bildlauf durchgeführt werden soll.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft wird von Einzeileneditierfeldern nicht bearbeitet.

EM_REPLACESEL

Beschreibung Diese Botschaft ersetzt die aktuelle Auswahl im Editierfeld durch neuen Text.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf einen nullterminierten String, der den Ersatztext enthält.

Rückgabewert Nicht verwendet.

EM_SETHANDLE

Beschreibung Diese Botschaft richtet den Textpuffer ein, der verwendet wird, um den Inhalt eines Editierfeldes zu speichern.

Parameter *wParam:* Enthält ein Handle des Puffers. Das Handle muß ein lokales Handle zu einer Position im Datensegment der Anwendung sein. Das Editierfeld verwendet diesen Puffer, um den aktuell angezeigten Text zu speichern, anstatt einen eigenen Puffer zu reservieren. Falls nötig, weist das Steuerelement diesen Puffer neu zu.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft wird von Einzeileneditierfeldern nicht verarbeitet.

Wenn *EM_SETHANDLE* verwendet wird, um den von einem Editierfeld verwendeten Textpuffer zu verändern, wird der vorherige Textpuffer nicht entfernt. Die Anwendung muß das vorherige Puffer-Handle mit Hilfe der Botschaft *EM_GETHANDLE* liefern und mit der Funktion *LocalFree* freigeben.

Ein Editierfeld weist den gegebenen Puffer immer dann neu zu, wenn es zusätzlichen Speicherplatz für Text benötigt oder wenn es genügend Text löscht, so daß zusätzlicher Speicherplatz nicht mehr benötigt wird. Eine Anwendung kann diese Botschaft nur an solche Editierfelder übergeben, deren Dialogfenster mit gesetztem Modus-Flag *DS_LOCALEDIT* erzeugt wurden.

EM_SETMODIFY

Beschreibung Diese Botschaft setzt das Veränderungsflag für ein gegebenes Editierfeld.

Parameter *wParam:* Enthält einen neuen Wert für das Veränderungsflag.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

EM_SETPASSWORDCHAR

Beschreibung Diese Botschaft setzt das Zeichen, das in einem mit dem Stil *ES_PASSWORD* erzeugten Editierfeld anstelle der eingegebenen Zeichen angezeigt wird.

Das voreingestellte Zeichen ist ein Stern (*).

Parameter *wParam:* Bestimmt das Zeichen, das anstelle des tatsächlich

vom Anwender eingegebenen Zeichens dargestellt werden soll. Ist *wParam* 0, werden die tatsächlich vom Anwender getippten Zeichen ausgegeben.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

EM_SETRECT

Beschreibung Diese Botschaft setzt das formatierende Rechteck für ein Editierfeld. Der Text wird neu formatiert und neu dargestellt, um dem veränderten Rechteck zu entsprechen.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf eine *RECT*-Datenstruktur, die die neuen Abmessungen des Rechtecks enthält.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft wird von Einzeileneditierfeldern nicht verarbeitet.

EM_SETRECTNP

Beschreibung Diese Botschaft setzt das formatierende Rechteck für ein Editierfeld. Der Text wird neu formatiert, um dem veränderten Rechteck zu entsprechen. Die Botschaft *EM_SETRECTNP* ähnelt der Botschaft *EM_SETRECT*, allerdings wird das Steuerelement nicht neu gezeichnet. Alle darauffolgenden Änderungen bewirken, daß das Steuerelement neu gezeichnet wird, um dem geänderten, formatierenden Rechteck zu entsprechen. Diese Botschaft wird verwendet, wenn das Feld erst nachträglich neu gezeichnet werden soll.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf eine *RECT*-Datenstruktur, die die neuen Abmessungen des Rechtecks enthält.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft wird von Einzeileneditierfeldern nicht verarbeitet.

EM_SETSEL

Beschreibung Diese Botschaft wählt alle Zeichen in einem Editierfeld aus, die innerhalb der Start- und Endzeichenpositionen liegen, die der Parameter *lParam* vorgibt.

Parameter *wParam:* Nicht verwendet.

LOWORD(lParam): Ist der Zeichenindex der Startposition.

HIWORD(lParam): Ist der Zeichenindex der Endposition.

Rückgabewert Nicht verwendet.

EM_SETTABSTOPS

Beschreibung Diese Botschaft setzt die Tabulatorpositionen in einem Mehrzeileneditierfeld.

Parameter *wParam:* Ist ein WORD, das die Anzahl der Tabulatorpositionen im Editierfeld enthält. Es kann auch die Werte 0 oder 1 annehmen.

lParam: Ist ein Zeiger auf das erste Element eines Arrays von Integerwerten, das die Tabulatorpositionen in Dialogeinheiten enthält. Eine Dialogeinheit ist ein horizontaler oder vertikaler Abstand. Eine horizontale Dialogeinheit ist gleich 1/4 und eine vertikale gleich 1/8 ihrer dazugehörigen Basiseinheit. Die Dialogbasiseinheiten werden aufgrund der Höhe und Breite der aktuellen Systemschrift gesetzt.

Die Funktion *GetDialogBaseUnits* gibt die aktuellen Dialogbasiseinheiten in Pixeln zurück. Die Tabulatorpositionen müssen in aufsteigender Reihenfolge sortiert werden, Rückwärtstabulatoren sind nicht erlaubt.

Rückgabewert Der Rückgabewert ist ungleich Null, wenn alle Tabulatoren gesetzt wurden, andernfalls Null.

Hinweis Sind *wParam* und *lParam* 0, werden voreingestellte Tabulatorpositionen im Abstand von 32 Dialogeinheiten gesetzt. Ist *wParam* 1, werden im Editierfeld Tabulatoren an jeder Position gesetzt, die einem Vielfachen von *lParam* entspricht.

EM_SETWORDBREAK

Beschreibung Diese Botschaft wird an ein Mehrzeileneditierfeld gesendet, um dieses darüber zu informieren, daß Windows die voreingestellte Wortumbruchfunktion durch eine von der Anwendung übergebene Wortumbruchfunktion ersetzt hat.

Eine Wortumbruchfunktion durchsucht einen Textpuffer, der an den Bildschirm zu sendenden Text enthält, nach dem ersten Wort, das nicht mehr in die aktuelle Bildschirmzeile paßt und setzt dieses Wort an den Anfang der nächsten Zeile auf dem Bildschirm.

Eine Wortumbruchfunktion definiert, an welchem Punkt Windows eine Textzeile in Mehrzeileneditierfeldern trennen soll. Üblicherweise ist dies bei einem Leerzeichen, das zwei

Worte trennt. Die von der Anwendung bereitgestellte Funktion kann einen Wortumbruch an einem Bindestrich oder einem anderen Zeichen als dem Leerzeichen definieren.

Parameter *wParam:* Nicht verwendet.

lParam: Die Adresse einer Prozedurinstanz. Diese Adresse muß mit der Funktion *MakeProcInstance* erzeugt werden. Die Wortumbruchfunktion kann folgendermaßen deklariert werden:

```
LPSTR WordBreakFunction(LPSTR EditText,
int CurrentWord, int EditTextCount)
```

WordBreakFunktion steht für den Namen der von der Anwendung bereitgestellten Callback-Funktion. Der Parameter *EditText* zeigt auf den Text eines Editierfeldes, der Parameter *CurrentWord* ist der Index des ersten Zeichens des aktuellen Worts. Der Parameter *EditTextCount* bestimmt die Anzahl der Bytes des Textes. Die Wortumbruchfunktion muß einen Zeiger auf den Anfang des nächsten Wortes im Text zurückgeben. Ist das aktuelle Wort das letzte Wort im Text, sollte die Funktion einen Zeiger auf das Zeichen unmittelbar nach dem letzten Textzeichen zurückgeben.

Rückgabewert Nicht verwendet.

Hinweis Die Standardwortumbruchfunktion von Windows definiert den Anfang des nächsten Worts als erstes Nicht-Leerzeichen nach einer Folge von Leerzeichen. Diese Botschaft ist nur auf Mehrzeileneditierfelder anwendbar.

EM_UNDO

Beschreibung Diese Botschaft macht den letzten Bearbeitungsvorgang im Editierfeld rückgängig. Verändert der Anwender das Editierfeld, wird die letzte Änderung in einem Widerrufen-Puffer gespeichert, der dynamisch mit den Anforderungen wächst. Ist für den Puffer nicht genügend Platz verfügbar, schlägt der Widerruf-Versuch fehl und das Editierfeld bleibt unverändert.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Bei erfolgreicher Ausführung der Undo-Operation ist der Rückgabewert ungleich 0, andernfalls 0. In diesem Fall bleibt das Editierfeld unverändert.

EN_-Benachrichtigungen

Hinweis Die EN_-Benachrichtigungen sind keine Windows-Botschaf-

ten im eigentlichen Sinn. Ihre Beschreibung finden Sie in Kapitel 4 »Windows-Stile und -Konstanten«.

LB_ADDSTRING

Beschreibung Diese Botschaft fügt einen String einer Liste hinzu. Ist die Liste unsortiert, wird der String am Ende der Liste angehängt, ist sie sortiert, wird der String in die Liste eingeordnet. Diese Botschaft löscht alle vorhandenen Listenauswahlen.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf den nullterminierten String, der hinzugefügt werden soll. Wenn die Liste besitzergezeichnet ist, aber ohne den Stil *LBS_HASSTRINGS* erzeugt wurde, ist *lParam* ein von der Anwendung bereitgestellter 32-Bit-Wert, der anstelle des Zeigers auf den String der Liste gespeichert wird.

Rückgabewert Der Rückgabewert ist der Index des Strings in der Liste. Der Rückgabewert ist *LB_ERR*, wenn ein Fehler auftritt, und *LB_ERRSPACE*, wenn nicht ausreichend Speicherplatz zur Verfügung steht, um den neuen String zu speichern.

Hinweis Wenn eine besitzergezeichnete Liste mit dem Stil *LBS_SORT*, aber ohne *LBS_HASSTRINGS* erstellt wurde, wird die Botschaft *WM_COMPAREITEM* ein- oder mehrmals an den Besitzer der Liste gesendet, so daß der neue Eintrag korrekt in der Liste plaziert werden kann.

LB_DELETESTRING

Beschreibung Diese Botschaft löscht einen String aus der Liste.

Parameter *wParam:* Enthält einen Index des Strings, der gelöscht werden soll.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Anzahl der verbleibenden Einträge in der Liste. Der Rückgabewert ist *LB_ERR*, wenn ein Fehler auftritt.

Hinweis Wenn die Liste besitzergezeichnet ist, aber ohne den Stil *LBS_HASSTRINGS* erstellt wurde, wird eine Botschaft *WM_DELETEITEM* an den Eigentümer der Liste gesendet, so daß die Anwendung zusätzliche, dem Eintrag durch den Parameter *lParam* der Botschaft *LB_ADDSTRING* oder *LB_INSERTSTRING* zugeordnete Daten freigeben kann.

LB_DIR

Beschreibung Diese Botschaft fügt eine Liste der Dateien aus dem aktuellen Verzeichnis zur Liste hinzu. Es werden nur Dateien mit den vom Parameter *wParam* bezeichneten Attributen sowie Dateien, die der vom Parameter *lParam* bezeichneten Dateivorgabe entsprechen, hinzugefügt.

Parameter *wParam:* Enthält einen DOS-Attributwert. Die DOS-Attribute werden bei der Funktion *DlgDirList* beschrieben.

lParam: Zeigt auf den nullterminierten Dateivorgabestring. Der String kann Stellvertreterzeichen enthalten (zum Beispiel *.*).

Rückgabewert Es wird der Index des letzten dargestellten Eintrags zurückgegeben. Der Rückgabewert ist *LB_ERR*, wenn ein Fehler aufritt und *LB_ERRSPACE*, wenn zum Speichern der neuen Strings nicht ausreichend Speicherplatz zur Verfügung steht.

Hinweis Der Rückgabewert der Botschaft *LB_DIR* ist um 1 kleiner als der Rückgabewert der Botschaft *LB_GETCOUNT*.

LB_FINDSTRING

Beschreibung Diese Botschaft findet in der Liste den ersten String, der mit dem bezeichneten Präfixstring übereinstimmt.

Parameter *wParam:* Enthält den Index des Eintrags vor dem ersten Eintrag, der untersucht werden soll. Wenn die Suche das Listenende erreicht, wird die Suche vom Beginn der Liste bis zu dem durch *wParam* bezeichneten Eintrag fortgesetzt. Ist *wParam* -1, wird die gesamte Liste von Anfang an durchsucht.

lParam: Zeigt auf den Präfixstring. Der String muß nullterminiert sein.

Rückgabewert Der Rückgabewert ist der Index des ersten entsprechenden Eintrags oder *LB_ERR*, wenn die Suche nicht erfolgreich war.

Hinweis Wenn die Liste besitzergezeichnet ist, aber ohne den Stil *LBS_HASSTRINGS* erstellt wurde, so gibt diese Botschaft den Index des Eintrags zurück, dessen zusätzlicher 32-Bit-Wert, übergeben als Parameter *lParam* der Botschaft *LB_ADDSTRING* oder *LB_INSERTSTRING*, mit dem Wert übereinstimmt, der von der Botschaft *LB_FINDSTRING* als Parameter *lParam* übergeben wurde.

LB_GETCOUNT

Beschreibung Diese Botschaft gibt die Anzahl der Einträge in einer Liste zurück.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Es wird die Anzahl der Einträge in der Liste zurückgegeben oder *LB_ERR*, wenn ein Fehler auftritt.

LB_GETCURSEL

Beschreibung Diese Botschaft gibt den Index des aktuell ausgewählten Eintrags zurück, falls ein solcher existiert.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist der Index des aktuell ausgewählten Eintrags. Er ist *LB_ERR*, wenn kein Eintrag ausgewählt oder die Liste für Mehrfachauswahl vorgesehen ist.

LB_GETHORIZONTALEXTENT

Beschreibung Diese Botschaft liefert die Breite in Pixeln, um die in der Liste ein Bildlauf durchgeführt werden kann, wenn die Liste horizontale Bildlaufleisten hat.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Breite des Bildlaufbereichs in der Liste in Pixeln.

Hinweis Um auf LB_GETHOTIZONTALEXTENT zu reagieren, muß die Liste mit dem Stil WS_HSCROLL erstellt worden sein.

LB_GETITEMDATA

Beschreibung Diese Botschaft liefert den von der Anwendung bereitgestellten 32-Bit-Wert, der dem bezeichneten Listeneintrag zugeordnet ist. Befindet sich der Eintrag in einer besitzergezeichneten Liste, die ohne den Stil *LBS_HASSTRINGS* erstellt wurde, war dieser 32-Bit-Wert im Parameter *lParam* der Botschaft *LB_-ADDSTRING* oder *LB_INSERTSTRING* enthalten, die den Eintrag zur Liste hinzugefügt hat. Andernfalls war es der Wert im Parameter *lParam* einer Botschaft *LB_SETITEMDATA.*

Parameter *wParam:* Enthält einen Index des Eintrags.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist der 32-Bit-Wert, der dem Eintrag zugeordnet ist oder *LB_ERR*, wenn ein Fehler auftritt.

LB_GETITEMRECT

Beschreibung Diese Botschaft liefert die Abmessungen des Rechtecks, das einen Listeneintrag so umgrenzt, wie er aktuell im Listenfenster dargestellt wird.

Parameter *wParam:* Enthält einen Index des Eintrags.

lParam: Ein Zeiger auf eine *RECT*-Datenstruktur, die die Client-Koordinaten des Rechtecks aufnimmt.

Rückgabewert Der Rückgabewert ist *LB_ERR*, wenn ein Fehler auftritt.

LB_GETSEL

Beschreibung Diese Botschaft gibt den Auswahlstatus eines Eintrags zurück.

Parameter *wParam:* Enthält den Index des Eintrags.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist eine positive Zahl, wenn ein Eintrag ausgewählt ist, andernfalls ist er 0. Der Rückgabewert ist *LB_ERR*, wenn ein Fehler auftritt.

LB_GETSELCOUNT

Beschreibung Diese Botschaft gibt die Gesamtzahl ausgewählter Einträge in einer Liste mit Mehrfachauswahlmöglichkeit zurück.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Anzahl der ausgewählten Einträge in einer Liste.

Wenn die Liste nur für Einzelauswahl vorgesehen ist, ist der Rückgabewert *LB_ERR*.

LB_GETSELITEMS

Beschreibung Diese Botschaft füllt einen Puffer mit einem Array mit Indizes der ausgewählten Einträge in einer Liste mit Mehrfachauswahl.

Parameter *wParam:* Bestimmt die maximale Anzahl von ausgewählten Einträgen, deren Indizes im Puffer gespeichert werden sollen.

lParam: Enthält einen Zeiger auf einen Puffer, der für die Anzahl von Integerwerten, die der Parameter *wParam* angibt, ausreichend groß sein muß.

Rückgabewert Der Rückgabewert ist die Anzahl der tatsächlich im Puffer gespeicherten Einträge.

Wenn die Liste eine Liste für Einzelauswahl ist, ist der Rückgabewert *LB_ERR*.

LB_GETTEXT

Beschreibung Diese Botschaft kopiert einen String aus der Liste in einen Puffer.

Parameter *wParam:* Enthält den Index des Strings, der kopiert werden soll.

lParam: Zeigt auf den Puffer, der den String aufnehmen soll. Der Puffer muß ausreichend groß sein, um sowohl den String als auch dessen abschließendes Nullzeichen aufnehmen zu können.

Rückgabewert Der Rückgabewert ist die Länge des Strings in Bytes, ohne Berücksichtigung des abschließenden Nullzeichens.

Der Rückgabewert ist *LB_ERR*, wenn der Parameter *wParam* keinen gültigen Index darstellt.

Hinweis Wenn die Liste besitzergezeichnet ist, aber ohne den Stil *LBS_HASSTRINGS* erstellt wurde, empfängt der Puffer, auf den der Parameter *lParam* der Botschaft zeigt, den 32-Bit-Wert, der dem Eintrag durch den Parameter *lParam* der Botschaften *LB_ADDSTRING* oder *LB_INSERTSTRING* zugeordnet wurde.

LB_GETTEXTLEN

Beschreibung Diese Botschaft gibt die Länge eines Strings in der Liste zurück.

Parameter *wParam:* Enthält den Index eines Eintrags.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist die Länge des Strings in Bytes ohne Berücksichtigung des abschließenden Nullzeichens, er ist *LB_ERR*, wenn ein Fehler auftritt.

LB_GETTOPINDEX

Beschreibung Diese Botschaft gibt den Index des ersten sichtbaren Eintrags in einer Liste zurück. Zu Beginn steht Eintrag 0 ganz oben in der Liste. Wird in der Liste ein Bildlauf durchgeführt, kann auch ein anderer Eintrag an oberster Stelle stehen.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Index des ersten sichtbaren Eintrags in einer Liste.

LB_INSERTSTRING

Beschreibung Diese Botschaft fügt einen String in eine Liste ein. Es wird keine Sortierung vorgenommen.

Parameter *wParam:* Enthält einen Index zu der Position, an der der String eingefügt werden soll. Wenn der Parameter *wParam* -1 ist, wird der String am Ende der Liste angefügt.

lParam: Zeigt auf den nullterminierten String, der eingefügt werden soll. Wenn die Liste besitzergezeichnet ist, aber ohne den Stil *LBS_HASSTRINGS* erzeugt wurde, ist der Parameter *lParam* ein von der Anwendung bereitgestellter 32-Bit-Wert, der von der Liste anstelle des Zeigers auf den String gespeichert wird.

Rückgabewert Der Rückgabewert ist der Index der Position, an der der String eingefügt wurde. Es wird *LB_ERR* zurückgegeben, wenn ein Fehler auftritt und *LB_ERRSPACE,* wenn zum Speichern des neuen Strings nicht genügend Speicherplatz zur Verfügung steht.

LB_RESETCONTENT

Beschreibung Diese Botschaft löscht alle Einträge aus einer Liste und gibt den gesamten für diese Strings reservierten Speicher frei.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Wenn die Liste besitzergezeichnet ist, aber ohne den Stil *LBS_HASSTRINGS* erstellt wurde, empfängt der Besitzer der Liste eine Botschaft *WM_DELETEITEM* für jeden Eintrag in der Liste.

LB_SELECTSTRINGS

Beschreibung Diese Botschaft setzt die aktuelle Auswahl auf den ersten String, der dem angegebenen Präfix entspricht.

Parameter *wParam:* Enthält den Index des Eintrags vor dem ersten zu untersuchenden Eintrag. Wenn die Suche das Listenende erreicht, wird sie vom Listenbeginn an bis zu dem von *wParam* bezeichneten Eintrag fortgesetzt. Wenn der Parameter *wParam* -1 ist, wird die gesamte Liste von Beginn an durchsucht.

lParam: Zeigt auf den Präfixstring. Der String muß nullterminiert sein.

Rückgabewert Der Rückgabewert ist der Index des ersten entsprechenden Eintrags oder *LB_ERR*, wenn ein Fehler auftritt.

Hinweis Diese Botschaft darf nicht für Listen verwendet werden, die für Mehrfachauswahl vorgesehen sind. Ein String wird nur ausgewählt, wenn seine Anfangsbuchstaben den Zeichen im Präfixstring entsprechen. Wenn die Liste besitzergezeichnet ist, aber ohne den Stil *LBS_HASSTRINGS* erstellt wurde, gibt diese Botschaft den Index des Eintrags zurück, dessen zusätzlicher 32-Bit-Wert, übergeben als Parameter *lParam* der Botschaft *LB_ADDSTRING* oder *LB_INSERTSTRING*, dem als Parameter *lParam* übergebenen Wert von *LB_FINDSTRING* entspricht.

LB_SELITEMRANGE

Beschreibung Diese Botschaft wählt einen oder mehrere aufeinanderfolgende Einträge in einer Mehrfachauswahlliste aus.

Parameter *wParam:* Bestimmt, wie die Auswahl gesetzt werden soll. Ist der Parameter *wParam* ungleich 0, werden die Einträge ausgewählt und hervorgehoben, ist *wParam* 0, wird die Hervorhebung zurückgenommen. Die Einträge sind dann nicht mehr ausgewählt.

LOWORD(lParam): Der Index des ersten auszuwählenden Eintrags.

HIWORD(lParam): Der Index des letzten auszuwählenden Eintrags.

Rückgabewert Der Rückgabewert ist *LB_ERR*, wenn ein Fehler auftritt.

Hinweis Diese Botschaft sollte nur für Mehrfachauswahlfenster verwendet werden.

LB_SETCOLUMNWIDTH

Beschreibung Diese Botschaft wird an eine Mehrspaltenliste gesendet, die mit dem Stil *LBS_MULTICOLUMN* erstellt wurde, um die Breite der Spalten in der Liste in Pixel zu setzen.

Parameter *wParam:* Bestimmt die Breite aller Spalten in Pixel.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

LB_SETCURSEL

Beschreibung Diese Botschaft wählt einen Eintrag aus und verschiebt ihn mittels Bildlauf ins Blickfeld, falls dies notwendig ist. Wenn

der neue Eintrag ausgewählt ist, löscht die Liste die Hervorhebung des vorher ausgewählten.

Parameter *wParam:* Enthält den Index des Eintrags, der ausgewählt werden soll.

Wenn *wParam* -1 ist, wird der Liste keine Auswahl zugeordnet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist *LB_ERR*, wenn *wParam* -1 oder ein ungültiger Index ist, andernfalls wird der Index des ausgewählten Eintrags zurückgegeben.

Hinweis Diese Botschaft sollte nur in Einzelauswahlfenstern verwendet werden. Sie kann nicht eingesetzt werden, um eine Auswahl in einem Mehrfachauswahlfenster zu setzen oder zu löschen.

LB_SETHORIZONTALEXTENT

Beschreibung Diese Botschaft stellt die Breite in Pixel ein, um die ein horizontaler Bildlauf durchgeführt werden kann.

Ist die Liste schmaler als dieser Wert, kann horizontaler Bildlauf in der Liste durchgeführt werden. Wenn die Liste breiter oder gleich diesem Wert ist, wird die horizontale Bildlaufleiste deaktiviert.

Parameter *wParam:* Bestimmt die Anzahl der Pixel, um die die Liste verschoben werden kann.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Um auf die Botschaft *LB_SETHORIZONTALEXTENT* reagieren zu können, muß die Liste mit dem Stil *WS_HSCROLL* erstellt worden sein.

LB_SETITEMDATA

Beschreibung Diese Botschaft setzt einen 32-Bit-Wert, der dem bezeichneten Eintrag in einer Liste zugeordnet ist.

Wenn der Eintrag in einer besitzergezeichneten Liste steht, die ohne den Stil *LBS_HASSTRINGS* erstellt wurde, ersetzt diese Botschaft den 32-Bit-Wert, der im Parameter *lParam* der Botschaft *LB_ADDSTRINGS* oder *LB_INSERTSTRING*, die den Eintrag zu der Liste hinzugefügt hat, enthalten war.

Parameter *wParam:* Enthält einen Index des Eintrags.

lParam: Enthält den neuen 32-Bit-Wert, der dem Eintrag zugeordnet werden soll.

Rückgabewert Der Rückgabewert ist *LB_ERR*, wenn ein Fehler auftritt.

LB_SETSEL

Beschreibung Diese Botschaft wählt in einer Liste mit Mehrfachauswahl einen Eintrag aus oder hebt dessen Auswahl auf.

Parameter *wParam:* Bestimmt, wie die Auswahl gesetzt werden soll. Ist der Parameter *wParam* ungleich 0, wird der Eintrag ausgewählt und hervorgehoben; ist er 0, wird die Hervorhebung zurückgenommen und der String ist nicht mehr ausgewählt.

lParam: Ist *lParam* -1, bezieht sich diese Botschaft auf alle Listeneinträge, andernfalls enthält der niederwertige Integerwert von *lParam* den Index des zu bearbeitenden Eintrags.

Rückgabewert Der Rückgabewert ist *LB_ERR*, wenn ein Fehler auftritt.

Hinweis Diese Botschaft sollte nur für Mehrfachauswahllisten verwendet werden.

LB_SETTABSTOPS

Beschreibung Diese Botschaft setzt die Tabulatoren in einer Liste.

Parameter *wParam:* Ein Integer, der die Anzahl der Tabulatoren in der Liste bestimmt.

lParam: Ein Zeiger auf das erste Element in einem Array aus Integerwerten, das die Tabulatorpositionen in Dialogeinheiten enthält. Eine Dialogeinheit ist ein horizontaler oder vertikaler Abstand. Eine horizontale Dialogeinheit ist gleich einem Viertel der zugehörigen Basiseinheit.

Bei der Berechnung der Dialogbasiseinheiten werden Breite und Höhe der aktuellen Systemschrift zugrundegelegt. Die Funktion *GetDialogBaseUnits* gibt die aktuellen Dialogbasiseinheiten in Pixel zurück. Die Tabulatoren müssen in aufsteigender Folge sortiert werden, Rückwärtstabulatoren sind nicht zulässig.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn alle Tabulatoren gesetzt wurden, andernfalls ist er 0.

Hinweis Wenn *wParam* und *lParam* 0 sind, beträgt der voreingestellte Tabulatorabstand zwei Dialogeinheiten. Wenn *wParam* 1 ist, verfügt das Editierfeld über Tabulatoren an jeder Position, die einem Vielfachen von *lParam* entspricht. Wenn *lParam* auf mehr als einen Wert zeigt, wird für jeden Wert in *lParam* ein Tabulator gesetzt, bis zu der Anzahl, die *wParam* vorgibt. Um auf die Botschaft *LB_SETTABSTOPS* reagieren zu können, muß die Liste mit dem Stil *LBS_USETABSTOPS* erstellt worden sein.

LB_SETTOPINDEX

Beschreibung Diese Botschaft macht das durch den Index bezeichnete Element zum ersten sichtbaren Eintrag in einer Liste.

Parameter *wParam:* Gibt den Index des Listeneintrags an.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist *LB_ERR*, wenn ein Fehler auftritt.

LBN_-Benachrichtigungen

Hinweis Die LBN_-Benachrichtigungen sind keine Windows-Botschaften im eigentlichen Sinn. Ihre Beschreibung finden Sie in Kapitel 4 »Windows-Stile und -Konstanten«.

WM_ACTIVATE

Beschreibung Diese Botschaft wird gesendet, wenn ein Fenster aktiviert oder deaktiviert wird.

Parameter *wParam:* Bestimmt den neuen Status des Fensters. Der Parameter *wParam* ist 0, wenn das Fenster nicht aktiv ist, andernfalls hat er einen der folgenden Werte:

Wert	Bedeutung
1	Das Fenster wurde nicht durch einen Mausklick, sondern z.B. durch einen Aufruf von *SetActiveWindow* oder durch Auswahl des Anwenders über die Tastaturschnittstelle aktiviert.
2	Das Fenster wurde durch einen Mausklick des Anwenders aktiviert. Es kann jede Maustaste verwendet werden.

HIWORD(lParam): Ist ungleich 0, wenn das Fenster zum Symbol verkleinert ist. Andernfalls ist er 0.

LOWORD(lParam): Wenn *wParam* 0 ist, ein Handle des zu aktivierenden, andernfalls des zu deaktivierenden Fensters.

Rückgabewert Nicht verwendet.

Hinweis Wenn das Fenster aktiviert wird und nicht verkleinert ist, ordnet die Funktion *DefWindowProc* als Standardaktion dem Fenster den Fokus zu.

WM_ACTIVATEAPP

Beschreibung Diese Botschaft wird übergeben, wenn das zu aktivierende Fenster einer anderen Anwendung angehört als das aktive Fenster. Die Botschaft wird sowohl an die Anwendung des zu

aktivierenden Fensters, als auch an die Anwendung des deswegen zu deaktivierenden Fensters übergeben.

Parameter *wParam:* Bestimmt, ob das Fenster aktiviert oder deaktiviert werden soll. Der Wert 0 zeigt an, daß Windows ein Fenster deaktivieren wird. Andere Werte bedeuten, daß ein Fenster aktiviert wird.

lParam: Das Task-Handle der anderen Anwendung.

Rückgabewert Nicht verwendet.

WM_ASKCBFORMATNAME

Beschreibung Diese Botschaft wird übergeben, wenn die Zwischenablage ein Daten-Handle für das Format *CF_OWNERDISPLAY* enthält (d.h. der gegenwärtige Besitzer der Zwischenablage sollte den Inhalt der Zwischenablage darstellen). Anschließend wird eine Kopie des Formatnamens angefordert.

Parameter *wParam:* Bestimmt die maximale Anzahl der zu kopierenden Bytes.

lParam: Zeigt auf den Puffer, der die Kopie des Formatnamens aufnehmen soll.

Rückgabewert Nicht verwendet.

Hinweis Der gegenwärtige Besitzer der Zwischenablage sollte beim Kopieren des Namens des Formats *CF_OWNERDISPLAY* die vorgegebene maximale Anzahl von Bytes nicht überschreiten. Format und Daten der Zwischenablage werden mit der Funktion *SetClipboardData* gesetzt.

WM_CANCELMODE

Beschreibung Diese Botschaft annuliert jeden Systemmodus, zum Beispiel den, der die Maus innerhalb einer Bildlaufleiste verfolgt, oder den, der ein Fenster bewegt. Windows übergibt die Botschaft *WM_CANCELMODE*, sobald eine Anwendung ein Meldungsfenster darstellt.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft teilt der Anwendung mit, daß Tastatur- und Mauseingabe auf das Meldungsfenster umgeleitet werden. Prozesse, die den Tastatur- bzw. Maustastenstatus und/oder die Mausposition überwachen, können nach dem Löschen des Meldungsfensters ungültig geworden sein.

WM_CHANGECBCHAIN

Beschreibung Diese Botschaft teilt dem ersten Fenster in der Viewer-Kette der Zwischenablage mit, daß ein Fenster aus der Kette entfernt wird.

Parameter *wParam:* Enthält das Handle des aus der Viewer-Kette der Zwischenablage zu entfernenden Fensters.

LOWORD(lParam): Ist ein Handle des Fensters, das dem zu entfernenden Fenster in der Viewer-Kette der Zwischenablage nachfolgt.

HIWORD(lParam): Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Jedes Fenster, das die Botschaft *WM_CHANGECBCHAIN* empfängt, sollte die Funktion *SendMessage* aufrufen, um die Botschaft an das folgende Fenster in der Viewer-Kette der Zwischenablage weiterzugeben.

Ist das zu entfernende Fenster das in der Kette folgende Fenster, rückt das durch den Parameter *LOWORD(lParam)* bezeichnete Fenster auf. Zwischenablagebotschaften werden an dieses Fenster weitergegeben. Fenster müssen sich selbst aus der Viewer-Kette der Zwischenablage entfernen, wenn sie die Botschaft *WM_DESTROY* empfangen.

Weitere Informationen finden sich in der Beschreibung der Botschaft *WM_DRAWCLIPBOARD*.

WM_CHAR

Beschreibung Diese Botschaft geht aus der Übersetzung einer *WM_KEYUP*- oder *WM_KEYDOWN*-Botschaft hervor. Sie enthält den Wert der gedrückten oder losgelassenen Taste.

Parameter *wParam:* Enthält den Wert der Taste.

LOWORD(lParam): Wiederholungszähler. Der Wert ergibt sich aus der Zeit, die der Anwender die Taste gedrückt hält und entspricht der ermittelten Anzahl von Tastenwiederholungen.

HIWORD(lParam): Die Bits von *HIWORD(lParam)* können folgende Werte annehmen:

Bit	Bedeutung
0 bis 7	Enthalten den Scancode der Taste (OEM-abhängiger Wert).
8	Ist 1, wenn die gedrückte Taste eine zusätzliche Taste (zum Beispiel Funktionstaste oder Taste des numerischen Tastenblocks) ist.

Bit	Bedeutung
9 bis 10	Nicht verwendet.
11 bis 12	Werden intern von Windows verwendet.
13	Ist 1, wenn gleichzeitig mit der Taste die Alt-Taste gedrückt wird.
14	Ist 1, wenn die Taste vor Übergabe dieser Botschaft bereits gedrückt war.
15	Ist 1, wenn die Taste losgelassen, 0, wenn sie gedrückt wird.

Rückgabewert Nicht verwendet.

Hinweis Da es keine notwendige Eins-zu-Eins-Entsprechung zwischen gedrückten Tasten und erzeugten Zeichenbotschaften gibt, ist der Inhalt der Bits 8 bis 15 des Parameters *HIWORD(lParam)* ohne Bedeutung für die Anwendung. Die Information dient ausschließlich der *WM_KEYUP*- oder *WM_KEYDOWN*-Botschaft, die der Übergabe der Zeichenbotschaft vorangeht.

Bei erweiterten IBM-Tastaturen mit 101 oder 102 Tasten gelten folgende Tasten als zusätzliche: die rechte Alt- und Ctrl-Taste (bzw. Strg) im Hauptbereich der Tastatur, die Tasten Ins, Del, Home, End, PgUp, PgDn und die Cursor-Tasten im Block zwischen dem Nummernblock und der alphanumerischen Tastatur sowie die Divisionstaste / und die Eingabetaste im Nummernblock. Auch andere Tastaturen unterstützen möglicherweise das Zusatztasten-Bit im Parameter *HIWORD(lParam)*. Nicht-System-Taste ist jede Taste, die gedrückt wird, wenn die Taste Alt nicht gedrückt wird.

Hat kein Fenster den Fokus, so werden die Botschaften *WM_SYSKEYDOWN*, *WM_SYSCHAR* und *WM_SYSKEYUP* anstelle von *WM_KEYDOWN*, *WM_CHAR* und *WM_KEYUP* übergeben.

Querverweis *WM_DEADCHAR*, *WM_SYSCHAR*, *WM_SYSDEADCHAR*

WM_CHARTOITEM

Beschreibung Diese Botschaft wird von einer Liste des Typs *LBS_WANTKEYBOARDINPUT* an ihren Besitzer als Reaktion auf eine *WM_CHAR*-Botschaft übergeben.

Parameter *wParam:* Enthält den Wert der vom Anwender gedrückten Taste.

LOWORD(lParam): Ein Handle zur Liste.

HIWORD(lParam): Die aktuelle Position des Caret.

Rückgabewert Der Rückgabewert bestimmt die Reaktion der Anwendung auf die Botschaft. Der Rückgabewert -2 zeigt an, daß die Anwen-

dung die Auswahl eines Eintrags vollständig bearbeitet hat und keine weiteren Aktionen in der Liste erwartet werden. Der Rückgabewert -1 zeigt an, daß die Liste die dem Tastendruck entsprechende Aktion ausführen soll. Ein Rückgabewert von 0 oder größer bestimmt den Index des Listeneintrags und zeigt an, daß die Liste die dem Tastendruck und dem bezeichneten Eintrag entsprechende Standardaktion ausführen soll.

Querverweis *WM_VKEYTOITEM*

WM_CHILDACTIVATE

Beschreibung Diese Botschaft wird an ein übergeordnetes Fenster übergeben, wenn die Funktion *SetWindowPos* das untergeordnete Fenster verschiebt.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_CLEAR

Beschreibung Diese Botschaft löscht die aktuelle Auswahl in einem Fenster.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_CLOSE

Beschreibung Diese Botschaft tritt beim Schließen eines Fensters auf.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Die Funktion *DefWindowProc* ruft die Funktion *DestroyWindow* auf, um das Fenster zu schließen. Vor Abarbeitung der Botschaft *WM_CLOSE* kann die Anwendung beim Anwender eine Bestätigung zur Schließung des Fensters abfragen. Stimmt der Anwender zu, wird die Funktion *DestroyWindow* aufgerufen.

WM_COMMAND

Beschreibung Diese Botschaft tritt auf, wenn der Anwender einen Menüeintrag auswählt, ein Steuerelement eine Botschaft an ihr überge-

ordnetes Fenster übergibt oder eine Akzelerator-Taste übersetzt wird.

Parameter *wParam:* Enthält den Menüeintrag, die ID eines Steuerelements oder die der Akzelerator-Taste.

LOWORD(lParam): Wenn *LOWORD(lParam)* 0 ist, stammt die Botschaft von einem Menü, *HIWORD(lParam)* wird in diesem Fall nicht verwendet. Ist *HIWORD(lParam)* ungleich 0, hängt die Interpretation des Werts von *HIWORD(lParam)* ab.

HIWORD(lParam): Ist *HIWORD(lParam)* 1 enthält *LOWORD(lParam)* die ID der Akzelerator-Taste. Andernfalls ist *LOWORD(lParam)* das Handle eines Steuerelements und *HIWORD(lParam)* ein Benachrichtigungscode, der an das übergeordnete Fenster des Steuerelements übergeben wird.

Rückgabewert Nicht verwendet.

Hinweis Eine Akzelerator-Taste ist für die Auswahl eines Eintrags aus dem Systemmenü definiert. Sie wird in eine *WM_SYSCOMMAND*-Botschaften übersetzt.

Wird ein Akzelerator-Tastendruck ausgeführt, der einem Menüeintrag entspricht, und ist das verwaltende Fenster des Menüs verkleinert, wird keine *WM_COMMAND*-Botschaft übergeben. Wenn aber der Akzelerator-Tastendruck keinem Menüeintrag des Fensters oder des Systemmenüs entspricht, wird eine *WM_COMMAND*-Botschaft übergeben, auch wenn das betreffende Fenster verkleinert ist.

Querverweis »BN_XXXXX für Aktionsschalter«, »LBN_XXXXX für Listen« in Kapitel 4, »Windows-Stile und -Konstanten«.

WM_COMPACTING

Beschreibung Diese Botschaft wird an alle Hauptfenster übergeben, sobald Windows feststellt, daß mehr als 12,5 Prozent der Systemzeit innerhalb eines 30 oder 60 Sekunden währenden Intervalls zur Neuanordnung des Speichers verwendet wird. Dies ist ein Zeichen für knappen Speicher. Erhält eine Anwendung diese Botschaft, sollte sie möglichst viel Speicher freigeben und dabei die eigenen Anforderungen sowie die Anzahl der unter Windows laufenden Anwendungen berücksichtigen. Deren Anzahl kann mit der Funktion *GetNumTasks* ermittelt werden.

Parameter *wParam:* Bestimmt den Anteil der CPU-Zeit, der für die Speicherverwaltung aufgewendet wird. 0x8000 entspricht 50 % der CPU-Zeit.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_COMPAREITEM

Beschreibung Diese Botschaft bestimmt die relative Position eines neuen Eintrags in einem sortierten, besitzergezeichneten Kombinationsfenster oder einer Liste.

Fügt die Anwendung einen neuen Eintrag hinzu, übergibt Windows diese Botschaft an den Besitzer des mit *CBS_SORT* oder *LBS_SORT* erstellten Kombinationsfensters oder der Liste.

Der Parameter *lParam* ist ein Zeiger auf eine *COMPAREITEMSTRUCT*-Datenstruktur, welche die Bezeichner und die von der Anwendung bereitgestellten Daten zweier Einträge des Kombinationsfensters oder der Liste enthält.

Sobald der Besitzer die Botschaft erhält, gibt er einen Wert zurück, der die Reihenfolge der Einträge anzeigt.

Normalerweise übergibt Windows diese Botschaft mehrmals, bevor die genaue Position des neuen Eintrags bestimmt werden kann.

Parameter *wParam:* Nicht verwendet.

lParam: Enthält einen Zeiger auf eine *COMPAREITEMSTRUCT*-Datenstruktur, die Bezeichner und anwendungsunterstützte Daten zweier Einträge in dem Kombinationsfenster oder der Liste enthält.

Rückgabewert Der Rückgabewert zeigt die relative Position zweier Einträge an. Folgende Werte sind möglich:

Wert	Bedeutung
-1	Eintrag 1 wird vor Eintrag 2 eingeordnet.
0	Eintrag 1 und 2 werden gleich eingeordnet.
1	Eintrag 1 wird nach Eintrag 2 eingeordnet.

WM_COPY

Beschreibung Diese Botschaft kopiert den aktuellen Auswahlbereich im *CF_TEXT*-Format in die Zwischenablage.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_CREATE

Beschreibung Diese Botschaft teilt der Fensterprozedur mit, daß eine Initialisierung möglich ist. Sie geht von der Funktion *CreateWindow* aus und wird vor Ende dieser Funktion und vor dem Öffnen des Fensters übergeben.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf eine Datenstruktur *CREATESTRUCT*, in der eine Kopie der an die Funktion *CreateWindow* übergebenen Parameter abgelegt ist.

Rückgabewert Nicht verwendet.

WM_CTLCOLOR

Beschreibung Diese Botschaft wird an das übergeordnete Fenster übergeben, wenn ein untergeordnetes Steuerelement oder ein Meldungsfenster gezeichnet werden soll. Dies ermöglicht es dem übergeordneten Fenster, Text- und Hintergrundfarben des untergeordneten Fensters unter Verwendung des in *wParam* vorgegebenen Handle des Bildschirmkontextes einzustellen.

Parameter *wParam:* Enthält das Handle des Bildschirmkontextes für das untergeordnete Fenster.

LOWORD(lParam): Ein Handle zum untergeordneten Fenster.

HIWORD(lParam): Eine der folgenden Konstanten, die den Typ des untergeordneten Fensters bestimmen:

Wert	Elementtyp
CTLCOLOR_BTN	Schalter.
CTLCOLOR_DLG	Dialogfenster.
CTLCOLOR_EDIT	Editierfeld.
CTLCOLOR_LISTBOX	Liste.
CTLCOLOR_MSGBOX	Meldungsfenster.
CTLCOLOR_SCROLLBAR	Bildlaufleiste.
CTLCOLOR_STATIC	Statisches Element.

Rückgabewert Nicht verwendet.

Hinweis Die Funktion *DefWindowProc* wählt die Standardsystemfarben. Wird die Botschaft *WM_CTLCOLOR* übergeben, muß die Anwendung den Ursprung des vorgesehenen Pinsels liefern.

Das geschieht, indem zuerst die Funktion *UnrealizeObject* für den Pinsel aufgerufen wird. Anschließend kann der Ursprung des Pinsels auf die obere linke Ecke des Fensters gesetzt werden.

Bearbeitet eine Anwendung die Botschaft *WM_CTLCOLOR*, so muß das Handle des für den Hintergrund vorgesehenen Pinsels zurückgegeben werden.

Die Übergabe eines ungültigen Handle kann einen Systemabsturz verursachen.

WM_CUT

Beschreibung Diese Botschaft übergibt die aktuelle Auswahl für die Zwischenablage im Format *CF_TEXT* und löscht die Auswahl anschließend aus dem Steuerelementfenster.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_DDE_ACK

Beschreibung Teilt einer Anwendung mit, daß eine andere DDE-Botschaft empfangen wurde.

Parameter *wParam*: Das Handle des die Botschaft sendenden Fensters.

lParam: War die empfangene Botschaft *WM_DDE_INITIATE*, enthält *LOWORD*(*lParam*) ein Atom, das die antwortende Anwendung benennt. *HIWORD*(*lParam*) enthält dann ein Atom mit dem Gegenstand der Übertragung.

War die empfangene Botschaft *WM_DDE_EXECUTE*, enthält *HIWORD*(*lParam*) einen Record, der den Status der Antwort bezeichnet. *HIWORD*(*lParam*) ist dann ein Handle auf den Dateneintrag, der die Befehlszeile enthält.

Bei allen anderen Botschaften ist *LOWORD*(*lParam*) eine Status-Struktur (DDEACK) und *HIWORD*(*lParam*) ein Atom, das den Dateneintrag bestimmt, für den die Antwort übergeben wird.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft muß mit der Funktion *SendMessage* übergeben werden. Der erste Parameter sollte dabei das Handle des die Botschaft empfangenden Fensters sein.

WM_DDE_ADVISE

Beschreibung Wird von der Client-Anwendung übergeben und fordert von der Server-Anwendung eine Aktualisierung an, wenn sich der Dateneintrag verändert.

Parameter *wParam*: Das Handle des sendenden Fensters.

LOWORD(*lParam*): Eine *DDEADVISE*-Struktur, die die Art des Datenaustausches bestimmt.

HIWORD(*lParam*): Ein Atom, das den angeforderten Dateneintrag bestimmt.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft muß mit der Funktion *PostMessage* übergeben werden. Der erste Parameter sollte das Handle des die Botschaft empfangenden Fensters sein.

WM_DDE_DATA

Beschreibung Wird von der Server-Anwendung übergeben, um den Wert eines Dateneintrags zu übertragen oder um der Client-Anwendung die Verfügbarkeit des Eintrags anzuzeigen.

Parameter *wParam*: Das Handle des sendenden Fensters.

LOWORD(lParam): Das Handle des globalen Speicherblocks, der die *DDEDATA*-Struktur mit den Daten enthält. *LOWORD (lParam)* ist 0, wenn die Botschaft nur zum Anzeigen einer Veränderung verwendet wird.

HIWORD(lParam): Ein Atom, das festlegt, welcher Dateneintrag übergeben wird.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft muß mit der Funktion *PostMessage* übergeben werden. Der erste Parameter sollte das Handle des die Botschaft empfangenden Fensters sein.

WM_DDE_EXECUTE

Beschreibung Wird von der Client-Anwendung gesendet, um eine Befehlsfolge zu übertragen, die von der Server-Anwendung bearbeitet werden soll.

Parameter *wParam*: Das Handle des sendenden Fensters.

LOWORD(lParam): Reserviert.

HIWORD(lParam): Ein Handle auf das globale Speicherobjekt, das die Befehle enthält.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft muß mit der Funktion *PostMessage* übergeben werden. Der erste Parameter sollte das Handle des die Botschaft empfangenden Fensters sein.

WM_DDE_INITIATE

Beschreibung Wird von der Client- oder der Server-Anwendung gesendet, um eine Übertragung zu initialisieren. Die antwortende Anwendung sollte eine *WM_DDE_ACK*-Botschaft übergeben.

Parameter *wParam*: Das Handle des sendenden Fensters.

LOWORD(lParam): Ein Atom, das den Namen der Anwendung festlegt, mit der die Übertragung stattfinden soll. Wenn *LOWORD(lParam)* 0 ist, kann die Übertragung mit jeder Anwendung durchgeführt werden.

HIWORD(lParam): Ein Atom, das den Gegenstand der gewünschten Übertragung bestimmt. Ist *HIWORD(lParam)* 0, kann sich die Übertragung auf jedes Thema beziehen.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft muß mit der Funktion *SendMessage* übergeben werden. Der erste Parameter sollte das Handle des die Botschaft empfangenden Fensters sein.

WM_DDE_POKE

Beschreibung Wird von der Client-Anwendung gesendet, um die Server-Anwendung zum Empfang dringender Daten zu veranlasssen.
Die Server-Anwendung antwortet mit einer *WM_DDE_ACK*-Botschaft.

Parameter *wParam*: Das Handle des sendenden Fensters.

LOWORD(lParam): Ein Handle auf eine *DDEPOKE*-Struktur.

HIWORD(lParam): Ein Atom, das den Dateneintrag bezeichnet.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft muß mit der Funktion *PostMessage* übergeben werden. Der erste Parameter sollte das Handle des die Botschaft empfangenden Fensters sein.

WM_DDE_REQUEST

Beschreibung Wird von der Client-Anwendung übergeben, um den Wert eines bestimmten Dateneintrags abzufragen.

Parameter *wParam*: Das Handle des sendenden Fensters.

LOWORD(lParam): Die Nummer eines Formats der Zwischenablage. Eine Beschreibung der Formate findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »CF_XXXX«.

HIWORD(lParam): Ein Atom, das den gewünschten Dateneintrag bezeichnet.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft muß mit der Funktion *PostMessage* übergeben werden. Der erste Parameter sollte das Handle des die Botschaft empfangenden Fensters sein.

WM_DDE_TERMINATE

Beschreibung Wird von einer Anwendung übergeben, um die Übertragung zu beenden.

Parameter *wParam*: Das Handle des sendenden Fensters.

lParam: Reserviert.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft muß mit der Funktion *PostMessage* übergeben werden. Der erste Parameter sollte das Handle des die Botschaft empfangenden Fensters sein.

WM_DDE_UNADVISE

Beschreibung Wird von der Client-Anwendung an die Server-Anwendung übergeben, um diese zu informieren, daß die Aktualisierung eines bestimmten Dateneintrags oder des Zwischenablageformats dieses Eintrags nicht länger nötig ist.

Parameter *wParam*: Das Handle des sendenden Fensters.

LOWORD(lParam): Die Nummer eines Formats der Zwischenablage. Eine Beschreibung der Formate findet sich in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »CF_XXXX«.

HIWORD(lParam): Ein Atom, das den Dateneintrag bezeichnet.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft muß mit der Funktion *PostMessage* übergeben werden. Der erste Parameter sollte das Handle des die Botschaft empfangenden Fensters sein.

WM_DEADCHAR

Beschreibung Diese Botschaft ergibt sich aus der Übersetzung einer *WM_-KEYUP*- und *WM_KEYDOWN*-Botschaft. Sie bestimmt den Wert einer toten Taste. Eine tote Taste ist beispielsweise der doppelte Punkt in den deutschen Umlauten (ä, ö, ü).

Parameter *wParam:* Enthält den Wert der toten Taste.

LOWORD(lParam): Wiederholungszähler. Der Wert ergibt sich aus der Zeit, die der Anwender die Taste gedrückt hält, und er entspricht der ermittelten Anzahl von Tastenwiederholungen.

HIWORD(lParam): Die Bits von *HIWORD(lParam)* können folgende Werte annehmen:

Bit	Bedeutung
0 bis 7	Enthalten den Scancode der Taste (OEM-abhängiger Wert).
8	Ist 1, wenn die gedrückte Taste eine zusätzliche Taste (zum Beispiel Funktionstaste oder Taste des numerischen Tastenblocks) ist.
9 bis 12	Werden entweder überhaupt nicht oder nur intern von Windows verwendet.
13	Ist 1, wenn gleichzeitig mit der Taste die Alt-Taste gedrückt wurde.
14	Ist 1, wenn die Taste vor Übergabe dieser Botschaft bereits gedrückt war.
15	Ist 1, wenn die Taste losgelassen, 0, wenn sie gedrückt wurde.

Rückgabewert Nicht verwendet.

Hinweis Die Botschaft *WM_DEADCHAR* kann von der Anwendung verwendet werden, um dem Anwender die Kontrolle über die tatsächlich betätigten Tasten zu ermöglichen. So kann ein Akzent an der aktuellen Caret-Position dargestellt werden, ohne daß das Caret bewegt werden müßte.

Da es keine notwendige Eins-zu-Eins-Entsprechung zwischen gedrückten Tasten und erzeugten Zeichenbotschaften gibt, ist der Inhalt der Bits 8 bis 15 des Parameters *HIWORD*(*lParam*) ohne Bedeutung für die Anwendung. Die Information dient ausschließlich der *WM_KEYUP*- oder *WM_KEYDOWN*-Botschaft, die der Übergabe der Zeichenbotschaften vorangeht.

Bei erweiterten IBM-Tastaturen mit 101 oder 102 Tasten gelten folgende Tasten als zusätzliche: die rechte Alt- und Ctrl-Taste im Hauptbereich der Tastatur, die Tasten Ins, Del, Home, End, PgUp, PgDn und die Cursor- Tasten im Block zwischen der numerischen und der alphanumerischen Tastatur sowie die Divisions- und die Eingabetaste im numerischen Tastenblock. Auch andere Tastaturen unterstützen eventuell das Zusatztasten-Bit in *HIWORD* (*lParam*).

WM_DELETEITEM

Beschreibung Diese Botschaft informiert den Besitzer einer besitzergezeichneten Liste oder eines Kombinationsfensters über die Entfernung eines Listeneintrags. Die Botschaft wird übergeben, wenn eine Liste oder ein Kombinationsfenster oder ein Eintrag aus ihnen durch eine der Botschaften *LB_DELETESTRING*, *LB_RESETCONTENT*, *CB_DELETESTRING* oder *CB_RESETCONTENT* entfernt wurde.

Parameter *wParam:* Nicht verwendet.

lParam: Enthält einen Zeiger auf eine *DELETEITEMSTRUCT*-Datenstruktur, die Informationen über einen gelöschten Listeneintrag beinhaltet.

Rückgabewert Nicht verwendet.

WM_DESTROY

Beschreibung Diese Botschaft meldet einem Fenster, daß es entfernt wird. Die Funktion *DestroyWindow* übergibt die Botschaft *WM_DESTROY* nach dessen Entfernung vom Bildschirm an das übergeordnete Fenster. Sie wird außerdem vor dem Entfernen eines Fensters an das übergeordnete Fenster übergeben.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Ist das zu löschende Fenster Teil der Viewer-Kette der Zwischenablage, die durch die Funktion *SetClipboardViewer* gesetzt wurde, muß das Fenster sich aus dieser Kette durch die Funktion *ChangeClipboardChain* vor der Rückkehr von der Botschaft *WM_DESTROY* löschen.

WM_DESTROYCLIPBOARD

Beschreibung Diese Botschaft wird an den Besitzer der Zwischenablage übergeben, wenn diese durch einen Aufruf der Funktion *EmptyClipboard* geleert wird.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_DEVMODECHANGE

Beschreibung Diese Botschaft wird nach Änderung einer Voreinstellung des Gerätemodus durch den Anwender an alle Hauptfenster übergeben.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf den in der Initialisierungsdatei WIN.INI festgelegten Gerätenamen.

Rückgabewert Nicht verwendet.

WM_DRAWCLIPBOARD

Beschreibung Diese Botschaft wird an das erste Fenster in der Viewer-Kette der Zwischenablage übergeben, sobald sich der Inhalt der Zwischenablage verändert. Eine Bearbeitung dieser Botschaft ist nur dann nötig, wenn die Anwendung durch einen Aufruf der Funktion *SetClipboardViewer* in die Viewer-Kette der Zwischenablage eingefügt wurde.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Jedes Fenster, das die Botschaft *WM_DRAWCLIPBOARD* erhält, sollte die Funktion *SendMessage* aufrufen, um sie an das nächste Fenster in der Viewer-Kette der Zwischenablage weiterzugeben. Das Handle des folgenden Fensters wird durch die Funktion *SetClipboardViewer* zurückgegeben, Änderungen können sich jedoch aus einer zwischenzeitlichen *WM_CHANGECBCHAIN*-Botschaft ergeben.

WM_DRAWITEM

Beschreibung Diese Botschaft teilt einem besitzergezeichneten Aktionsschalter, einem Kombinationsfenster, einer Liste oder einem Menü mit, daß ein sichtbarer Bestandteil des Steuerelementes verändert wurde. Das Feld *itemAction* in der *DRAWITEMSTRUCT*-Struktur bestimmt die durchzuführende Darstellungsoperation. Diese Daten erlauben es dem Besitzer der Steuerelemente, die Darstellungsoperation festzulegen.

Parameter *wParam:* Nicht verwendet.

lParam: Enthält einen Zeiger auf eine *DRAWITEMSTRUCT*-Datenstruktur, die Informationen über Gegenstand und Typ der Darstellungsoperation enthält.

Rückgabewert Nicht verwendet.

Hinweis Vor Abschluß der Bearbeitung dieser Botschaft sollte die Anwendung alle Objekte des Gerätekontextes, die vom Feld *hDC* der *DRAWITEMSTRUCT*-Datenstruktur unterstützt werden, wiederherstellen.

WM_ENABLE

Beschreibung Diese Botschaft wird nach Aktivierung oder Deaktivierung eines Fensters übergeben.

Parameter *wParam:* Bestimmt, ob das Fenster aktiviert oder deaktiviert wurde. Der Parameter *wParam* ist ungleich 0, wenn das Fenster aktiviert wurde. Andernfalls hat er den Wert 0.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_ENDSESSION

Beschreibung Diese Botschaft wird an eine Anwendung übergeben, die auf die Botschaft *WM_QUERYENDSESSION* einen Wert ungleich 0 zurückgegeben hat, um ihr mitzuteilen, daß die Arbeitssitzung beendet wird.

Parameter *wParam:* Bestimmt, ob die Sitzung beendet wird. In diesem Fall ist der Wert ungleich 0, andernfalls 0.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Hat der Parameter *wParam* einen Wert ungleich 0, kann Windows die Arbeitssitzung jederzeit nach Bearbeitung dieser Botschaft durch die Anwendungen beenden. Eine Anwendung sollte deshalb alle für die Beendigung notwendigen Aufgaben durchführen, bevor sie von der Bearbeitung dieser Botschaft zurückkehrt.

Ein Aufruf der Funktionen *DestroyWindow* oder *PostQuitMessage* ist nicht nötig.

WM_ENTERIDLE

Beschreibung Diese Botschaft teilt der Hauptfensterfunktion einer Anwendung mit, daß ein modales Dialogfenster oder ein Menü keine Aufgaben mehr hat. Dieser Status wird erreicht, wenn sich nach der Bearbeitung einer oder mehrerer Botschaften keine weiteren zu bearbeitenden Botschaften in der Warteschlange des modalen Dialogfensters bzw. des Menüs befinden.

Parameter *wParam:* Legt fest, ob die Botschaft Resultat eines dargestellten Dialogfensters oder Menüs ist. Der Parameter hat einen der folgenden Werte:

Wert	Bedeutung
MSGF_DIALOGBOX	Das System liegt wegen der Darstellung eines Dialogfensters still.
MSGF_MENU	Das System liegt wegen der Darstellung eines Menüs still.

LOWORD(lParam): Ein Handle zum Dialogfenster oder Menü, wenn *wParam* den Wert *MSGF_DIALOGBOX* oder *MSGF_MENU* hat.

HIWORD(lParam): Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Das System liegt still, weil ein modales Dialogfenster oder ein dargestelltes Menü keine Botschaften in der Warteschlange hat, nachdem mindestens eine vorhergehende Botschaft bearbeitet wurde.

Der Standardrückgabewert der Funktion *DefWindowProc* ist 0.

WM_ERASEBKGND

Beschreibung Diese Botschaft wird übergeben, wenn der Hintergrund eines Fensters gelöscht werden muß, weil z.B. die Größe eines Fensters verändert wird. Mit ihrer Hilfe wird ein ungültig gewordener Bereich für das Zeichnen vorbereitet.

Parameter *wParam:* Enthält des Handle des Gerätekontextes.

lParam: Nicht verwendet.

Rückgabewert Bearbeitet eine Anwendung die Botschaft und löscht sie den Hintergrund des Fensters, sollte sie einen Wert ungleich 0 zurückgeben. Andernfalls sollte der Rückgabewert 0 sein.

Hinweis Die Standardaktion der Funktion *DefWindowProc* ist das Löschen des Hintergrundes unter Verwendung des Hintergrundpinsels der Klassenstruktur. Ist der Hintergrundpinsel der Klasse 0, sollte die Anwendung den Ursprung des Pinsels ausrichten, der für das Fenster verwendet werden soll.

Sie sollte die Funktion *UnrealizeObject* aufrufen, einen Pinsel auswählen und dann den Hintergrund unter Verwendung dieses Pinsels löschen. Windows geht dabei vom Abbildungsmoduls *MM_TEXT* aus.

Arbeitet der Bildschirmkontext mit einem anderen Abbildungsmodus, geht der gelöschte Bereich möglicherweise über den sichtbaren Bereich des Client-Bereiches hinaus.

WM_FONTCHANGE

Beschreibung Diese Botschaft tritt auf, wenn sich der Bestand der Schriftressourcen verändert. Jede Anwendung, die Schriften zum System hinzufügt oder aus ihm entfernt (z.B. mit Hilfe der Funktionen *AddFontResource* oder *RemoveFontResource*), sollte diese Botschaft an alle Hauptfenster übergeben.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Um die Botschaft *WM_FONTCHANGE* an alle Hauptfenster zu übergeben, kann die Anwendung die Funktion *SendMessage* aufrufen.

WM_GETDLGCODE

Beschreibung Diese Botschaft wird von Windows an die zu einem Steuerelement gehörende Eingabeprozedur übergeben. Üblicherweise gibt Windows alle Eingaben der Cursor-Tasten oder der Tab-Taste an das Steuerelement ab. Eine Anwendung kann die Kontrolle einer bestimmten Art der Eingabe sowie deren Bearbeitung durch Beantworten der Botschaft *WM_GETDLGCODE* übernehmen.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert besteht aus mindestens einem der folgenden Werte, die den Typ der durch die Anwendung überwachten Eingabe festlegen:

Wert	Bedeutung
DLGC_DEFPUSHBUTTON	Standardaktionsschalter.
DLGC_HASSETSEL	*EM_SETSEL*-Botschaften.
DLGC_PUSHBUTTON	Aktionsschalter.
DLGC_RADIOBUTTON	Schaltfeld.
DLGC_WANTALLKEYS	Alle Tastatureingaben.
DLGC_WANTARROWS	Cursor-Tasten.
DLGC_WANTCHARS	*WM_CHAR*-Botschaften.
DLGC_WANTMESSAGE	Alle Tastatureingaben (die Anwendung leitet die Botschaft an das Steuerelement weiter).
DLGC_WANTTAB	Tab-Taste.

Hinweis Der Standardrückgabewert von *DefWindowProc* ist 0. Obwohl die Funktion *DefWindowProc* als Reaktion auf die Botschaft *WM_GETDLGCODE* immer 0 zurückgibt, stellt der Rückgabewert der Fensterfunktionen der vordefinierten Steuerelementklassen einen der jeweiligen Klasse entsprechenden Code dar.

Die *WM_GETDLGCODE*-Botschaft und die Rückgabewerte sind nur bei der Arbeit mit anwenderdefinierten Dialogsteuerelementen und unterklassifizierten Standardsteuerelementen hilfreich.

WM_GETFONT

Beschreibung Diese Botschaft liefert die aktuell zur Textdarstellung in einem Dialogfenster verwendete Schrift.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert entspricht dem Handle der Schrift, mit dem das Steuerelement arbeitet. Er ist 0, wenn die Systemschrift verwendet wird.

WM_GETMINMAXINFO

Beschreibung Diese Botschaft wird übergeben, wenn Windows die Maximalgröße eines Fensters, seine minimale oder maximale Tracking-Größe oder dessen Position im vergrößerten Zustand benötigt. Das Vollbild eines Fensters ist erreicht, wenn seine Grenzen soweit als möglich ausgedehnt worden sind.

Die maximale Tracking-Größe entspricht der größtmöglichen Fenstergröße, die durch Größeneinstellung mit Hilfe der Fensterränder erreicht werden kann. Die minimale Tracking-Größe entspricht der kleinstmöglichen Fenstergröße, die mit Hilfe der Fensterränder erreicht werden kann.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf ein Array mit fünf *POINT*-Strukturen:

Point	Beschreibung
lParam[*0*]	Zur internen Verwendung durch Windows.
lParam[*1*]	Die Maximalgröße, sie ist die Standardeinstellung. Die Breite ergibt sich aus der Formel (*SM_CXSCREEN* + (2 * *SM_CXFRAME*)), die Höhe aus (*SM_CYSCREEN* + (2 * *SM_CYFRAME*)).
lParam[*2*]	Die Position der oberen linken Ecke des vergrößerten Fensters in Bildschirmkoordinaten. Der Vorgabewert für x ist *SM_CXFRAME*, für y *SM_CYFRAME*.
lParam[*3*]	Die kleinstmögliche Tracking-Größe, Vorgabe ist die Größe des Symbols. Die Breite ist *SM_CXMINTRACK*, die Höhe *SM_CYMINTRACK*.
lParam[*4*]	Die größtmögliche Tracking-Größe, deren Vorgabewert kleiner als die Bildschirmgröße ist. Die Breite ergibt sich aus der Formel (*SM_CXSCREEN* + (2 * *SM_CXFRAME*)), die Höhe aus (*SM_CYSCREEN* + (2 * *SM_CYFRAME*)).

Rückgabewert Die Elemente 1 bis 4 des Arrays, auf das *lParam* zeigt, können nach Belieben geändert werden.

Hinweis Das Array enthält vor der Übergabe der Botschaft *WM_GETMINMAXINFO* durch Windows die Standardwerte für jeden Punkt. Mit Hilfe dieser Botschaft können die Standardwerte durch die Anwendung verändert werden.

WM_GETTEXT

Beschreibung Diese Botschaft wird verwendet, um den zu einem Fenster gehörenden Text zu kopieren. Bei Editierfeldern und Editierfeldern in Kombinationsfenstern entspricht der zu kopierende Text dem Inhalt der Felder. Bei Schaltern entspricht der Text dem Namen des Schalters. In Listen entspricht der Text dem ausgewählten Eintrag. In anderen Fenstern entspricht der Text der Überschrift.

Parameter *wParam:* Bestimmt die maximale Anzahl der zu kopierenden Bytes, einschließlich des abschließenden Zeichens.

lParam: Zeigt auf den Puffer, der den Text aufnehmen soll. Er muß ausreichend groß sein, um die in *wParam* bestimmte Zeichenzahl aufzunehmen.

Rückgabewert Der Rückgabewert entspricht der Anzahl der kopierten Bytes, einschließlich des Null-Zeichens. Er ist *LB_ERR*, wenn kein Eintrag ausgewählt war. Hat das Kombinationsfenster kein Editierfeld, ist er *CB_ERR*.

Hinweis Weitere Informationen finden sich in den Beschreibungen der Botschaften *WM_GETTEXTLENGTH* und *WM_SETTEXT*.

WM_GETTEXTLENGTH

Beschreibung Diese Botschaft wird zur Ermittlung der Länge eines zu einem Fenster gehörenden Textes verwendet. In diese Angabe in Bytes ist die abschließende Null nicht eingeschlossen. Bei Editierfeldern und Editierfeldern in Kombinationsfenstern ist der Text der Inhalt des Steuerelements. In Listen entspricht der Text dem ausgewählten Eintrag, bei Schaltern entspricht er dem Schalternamen und in anderen Fenstern der Überschrift.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert entspricht der Länge des gegebenen Textes, ohne Berücksichtigung des abschließenden Nullzeichens.

Hinweis Weitere Informationen finden sich in der Beschreibung der Botschaft *WM_GETTEXT*.

WM_HSCROLL

Beschreibung Diese Botschaft wird übergeben, wenn der Anwender die horizontale Bildlaufleiste anklickt.

Parameter *wParam:* Enthält einen Bildlaufleistencode, der die Bildlaufwünsche des Anwenders angibt. Folgende Werte sind möglich:

Wert	Bedeutung
SB_BOTTOM	Bildlauf nach unten rechts.
SB_ENDSCROLL	Ende des Bildlaufs.
SB_LINEDOWN	Bildlauf eine Zeile nach unten.
SB_LINEUP	Bildlauf eine Zeile nach oben.
SB_PAGEDOWN	Bildlauf eine Seite nach unten.
SB_PAGEUP	Bildlauf eine Seite nach oben.
SB_THUMBPOSITION	Bildlauf zu einer absoluten Position. Die aktuelle Position wird in *LOWORD(lParam)* bereitgestellt.
SB_THUMBTRACK	Mauszug zur festgelegten Position. Die aktuelle Position wird in *LOWORD(lParam)* bereitgestellt.
SB_TOP	Bildlauf nach oben links.

Parameter *LOWORD(lParam):* Die aktuelle Position des Positionszeigers.

HIWORD(lParam): Bestimmt das Handle des Steuerelements der Bildlaufleiste. *HIWORD(lParam)* ist 0, wenn die Bildlaufleiste mit dem Stil *WS_HSCROLL* erstellt wurde. Wird die Botschaft infolge eines Klicks des Anwenders auf die Bildlaufleiste eines Pop-up-Fensters übergeben, wird *HIWORD(lParam)* nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Die Botschaft *SB_THUMBTRACK* wird üblicherweise von Anwendungen verwendet, die auf Verschieben der Positionsmarke reagieren.

Führt eine Anwendung einen Bildlauf in einem Fenster durch, muß sie die Positionsmarke durch die Funktion *SetScrollPos* zurücksetzen.

WM_HSCROLLCLIPBOARD

Beschreibung Diese Botschaft wird übergeben, wenn die Zwischenablage ein Daten-Handle auf das Format *CF_OWNERDISPLAY* enthält (der Besitzer der Zwischenablage sollte die enthaltenen Daten darstellen) und ein Ereignis an der horizontalen Bildlaufleiste der Zwischenablage vorliegt.

Parameter *wParam:* Enthält ein Handle zum Fenster der Zwischenablage.

LOWORD(lParam): Enthält einen der folgenden Bildlaufleistencodes:

Wert	Bedeutung
SB_BOTTOM	Bildlauf nach unten rechts.
SB_ENDSCROLL	Ende des Bildlaufs.
SB_LINEDOWN	Bildlauf eine Zeile nach unten.
SB_LINEUP	Bildlauf eine Zeile nach oben.
SB_PAGEDOWN	Bildlauf eine Seite nach unten.
SB_PAGEUP	Bildlauf eine Seite nach oben.
SB_THUMBPOSITION	Bildlauf zu einer absoluten Position.
SB_TOP	Bildlauf nach oben links.

HIWORD(lParam): Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Der Besitzer der Zwischenablage sollte die Funktion *InvalidateRect* verwenden oder wie gewünscht neu darstellen. Die Positionsmarke sollte zurückgesetzt werden.

Folgende Meldungen werden übergeben, wenn das Format der Zwischenablage *CF_OWNERDISPLAY* ist: *WM_ASKCBFORMATNAME*, *WM_HSCROLLCLIPBOARD*, *WM_PAINTCLIPBOARD* und *WM_VSCROLLCLIPBOARD*.

Daten und Format der Zwischenablage werden mit der Funktion *SetClipboardData* gesetzt.

WM_ICONERASEBKGND

Beschreibung Diese Botschaft wird an ein verkleinertes Fenster (Symbol) übergeben, wenn der Hintergrund vor Zeichnung des Symbols ausgemalt werden muß. Das Fenster empfängt diese Botschaft nur, wenn ein Klassensymbol für dieses Fenster definiert wurde, andernfalls wird *WM_ERASEBKGND* übergeben.

Die Übergabe dieser Botschaft an die Funktion *DefWindowProc* ermöglicht es Windows, den Hintergrund des Symbols mit dem Hintergrundpinsel des übergeordneten Fensters auszufüllen.

Parameter *wParam:* Enthält das Handle des Gerätekontextes des Symbols.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_INITDIALOG

Beschreibung Diese Botschaft wird unmittelbar vor der Darstellung eines Dialogfensters übergeben. Durch Abarbeitung dieser Botschaft

kann die Anwendung beliebige Initialisierungen vor dem Anzeigen des Dialogfensters durchführen.

Parameter *wParam:* Bezeichnet das erste Steuerelement des Dialogfensters, dem der Fokus zugeordnet werden kann. Dieses ist im allgemeinen das erste Element des Stils *WS_TABSTOP*.

lParam: Ist der Wert, der als Parameter *InitParam* übergeben wird, wenn das Dialogfenster durch eine der folgenden Funktionen erzeugt wird: *CreateDialogIndirectParam*, *CreateDialogParam*, *DialogBoxIndirectParam* und *DialogBoxParam*. Wurde das Dialogfenster mit *CreateDialog*, *CreateDialogIndirect*, *DialogBox* oder *DialogBoxIndirect* erzeugt, ist *lParam* 0. In allen anderen Fällen wird *lParam* nicht verwendet.

Rückgabewert Wird der Fokus einem der Steuerelemente eines Dialogfensters zugeordnet, kann der Rückgabewert 0 sein. Andernfalls muß er ungleich 0 sein.

Hinweis Gibt die Anwendung in Reaktion auf die Botschaft *WM_INITDIALOG* einen Wert ungleich 0 zurück, setzt Windows den Fokus auf den durch das Handle des Parameters *wParam* bezeichneten Eintrag.

WM_INITMENU

Beschreibung Diese Botschaft ist die Aufforderung zum Initialisieren eines Menüs. Sie tritt auf, wenn der Anwender eine Menüleiste anklickt oder eine Menütaste betätigt. Windows übergibt diese Botschaft, bevor es ein Menü darstellt. Dies ermöglicht es der Anwendung, den Status der Menüeinträge vorab einzustellen.

Parameter *wParam:* Enthält das Menü-Handle des zu initialisierenden Menüs.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Die *WM_INITMENU*-Botschaft wird nur beim ersten Zugriff auf ein Menü übergeben, es wird nur eine Botschaft für jeden Zugriff erzeugt. Das Bewegen der Maus über die Einträge mit gedrückter Maustaste erzeugt also keine neuen Botschaften.

Die Botschaft übergibt keine Informationen über die Menüeinträge.

WM_INITMENUPOPUP

Beschreibung Diese Botschaft wird unmittelbar vor Darstellung eines Pop-up-Menüs übergeben. Die Bearbeitung der Botschaft ermöglicht es der Anwendung, den Status der Pop-up-Menüeinträge

vorab und ohne Auswirkung auf den gesamten Menüstatus zu verändern.

Parameter *wParam:* Enthält das Menü-Handle des Pop-up-Menüs.

LOWORD(lParam): Der Index des Pop-up-Menüs im Hauptmenü.

HIWORD(lParam): Ist ungleich 0, wenn das Pop-up-Menü das Systemmenü ist. Andernfalls ist *HIWORD(lParam)* 0.

Rückgabewert Nicht verwendet.

WM_KEYDOWN

Beschreibung Diese Botschaft wird bei Betätigung einer Standardtaste übergeben. Eine Standardtaste (Nicht-System-Taste) ist jede Taste, die nicht zusammen mit der Alt-Taste gedrückt wird, sowie jede Taste, die betätigt wird, während ein Fenster den Fokus besitzt.

Parameter *wParam:* Bestimmt den virtuellen Tastencode der vorgegebenen Taste.

LOWORD(lParam): Wiederholungszähler. Der Wert ergibt sich aus der Zeit, die der Anwender die Taste gedrückt hält und entspricht der ermittelten Anzahl von Tastenwiederholungen.

HIWORD(lParam): Die Bits von *HIWORD(lParam)* können folgende Werte annehmen:

Bit	Bedeutung
0 bis 7	Enthalten den Scancode der Taste (OEM-abhängiger Wert).
8	Ist 1, wenn die gedrückte Taste eine zusätzliche Taste (zum Beispiel Funktionstaste oder Taste des numerischen Tastenblocks) ist.
9 bis 12	Werden entweder überhaupt nicht oder nur intern von Windows verwendet.
13	Ist 1, wenn gleichzeitig mit der Taste die Alt-Taste gedrückt wurde.
14	Ist 1, wenn die Taste vor Übergabe dieser Botschaft bereits gedrückt war.
15	Ist 1, wenn die Taste losgelassen, 0, wenn sie gedrückt wurde.

Rückgabewert Nicht verwendet.

Hinweis Aufgrund der automatischen Tastenwiederholung können einer *WM_KEYUP*-Botschaft mehr als eine *WM_KEYDOWN*-Botschaft vorangehen. Das Bit 14 von *HIWORD(lParam)* kann

verwendet werden, um festzustellen, ob es sich um die erste oder eine wiederholte *WM_KEYDOWN*-Botschaft handelt.

Bei erweiterten IBM-Tastaturen mit 101 oder 102 Tasten gelten folgende Tasten als zusätzliche: die rechte Alt- und Ctrl- (bzw. Strg)-Taste im Hauptbereich der Tastatur, die Tasten Ins, Del, Home, End, PgUp, PgDn und die Cursor-Tasten im Block zwischen dem Nummernblock und der alphanumerischen Tastatur sowie die Divisions- / und die Eingabetaste im Nummernblock. Auch andere Tastaturen unterstützen möglicherweise das Zusatztasten-Bit im Parameter *HIWORD*(*lParam*).

Hat kein Fenster den Fokus, so werden die Botschaften *WM_SYSKEYDOWN*, *WM_SYSCHAR* und *WM_SYSKEYUP* anstelle von *WM_KEYDOWN*, *WM_CHAR* und *WM_KEYUP* übergeben. Vergleichen Sie auch die Beschreibung der Botschaften *WM_DEADCHAR*, *WM_SYSCHAR* und *WM_SYSDEADCHAR*.

WM_KEYUP

Beschreibung Diese Botschaft wird beim Loslassen einer Standardtaste übergeben. Eine Standardtaste (Nicht-System-Taste) ist jede Taste, die nicht zusammen mit der Alt-Taste gedrückt wird, sowie jede Taste, die betätigt wird, während ein Fenster den Fokus besitzt.

Parameter *wParam:* Bestimmt den virtuellen Tastencode der Taste.

LOWORD(*lParam*): Wiederholungszähler. Der Wert ergibt sich aus der Zeit, die der Anwender die Taste gedrückt hält und entspricht der ermittelten Anzahl von Tastenwiederholungen.

HIWORD(*lParam*): Die Bits von *HIWORD*(*lParam*) können folgende Werte annehmen:

Bit	Bedeutung
0 bis 7	Enthalten den Scancode der Taste (OEM-abhängiger Wert).
8	Ist 1, wenn die gedrückte Taste eine zusätzliche Taste (zum Beispiel Funktionstaste oder Taste des numerischen Tastenblocks) ist.
9 bis 12	Werden entweder überhaupt nicht oder nur intern von Windows verwendet.
13	Ist 1, wenn gleichzeitig mit der Taste die Alt-Taste gedrückt wurde.
14	Ist 1, wenn die Taste vor Übergabe dieser Botschaft bereits gedrückt war.
15	Ist 1, wenn die Taste losgelassen, 0, wenn sie gedrückt wurde.

Rückgabewert Nicht verwendet.

Hinweis Bei erweiterten IBM-Tastaturen mit 101 oder 102 Tasten gelten folgende Tasten als zusätzliche: die rechte Alt- und Ctrl- (bzw. Strg-)Taste im Hauptbereich der Tastatur, die Tasten Ins, Del, Home, End, PgUp, PgDown und die Cursor-Tasten im Block zwischen dem Nummernblock und der alphanumerischen Tastatur sowie die Divisions- und die Eingabetaste im Nummernblock.

Auch andere Tastaturen unterstützen möglicherweise das Zusatztasten-Bit im Parameter *HIWORD(lParam)*.

WM_KILLFOCUS

Beschreibung Diese Botschaft wird an ein Fenster übergeben, unmittelbar bevor es den Fokus verliert.

Parameter *wParam:* Enthält das Handle des Fensters, das den Fokus besitzt.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Stellt eine Anwendung ein Caret dar, sollte dieses jetzt entfernt und freigegeben werden. *wParam* kann auch den Wert 0 haben.

WM_LBUTTONDBLCLK

Beschreibung Diese Botschaft tritt auf, wenn der Anwender mit der linken Maustaste doppelklickt.

Parameter *wParam:* Enthält einen Wert, der anzeigt, welche virtuellen Tasten außerdem gerade gedrückt werden. Möglich ist jede Kombination der in der folgenden Tabelle aufgeführten Werte:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt wird.
MK_LBUTTON	Gesetzt, wenn die linke Maustaste gedrückt wird.
MK_MBUTTON	Gesetzt, wenn die mittlere Maustaste gedrückt wird.
MK_RBUTTON	Gesetzt, wenn die rechte Maustaste gedrückt wird.
MK_SHIFT	Gesetzt, wenn die Shift-Taste gedrückt wird.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters. Nur Fenster, deren Fensterklasse den Stil *CS_DBLCLKS* hat, können Doppelklickbotschaften empfangen.

Windows erzeugt eine Doppelklickbotschaft, wenn eine Maustaste gedrückt, losgelassen und innerhalb der vom System vorgegebenen Doppelklickzeit ein zweites Mal betätigt wird. Ein Doppelklick erzeugt vier Botschaften: Drücken der Taste, Loslassen der Taste, Doppelklickbotschaft, erneutes Loslassen der Taste.

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_LBUTTONDOWN

Beschreibung Diese Botschaft tritt auf, wenn der Anwender die linke Maustaste drückt.

Parameter *wParam:* Enthält einen Wert, der den Status verschiedener virtueller Tasten anzeigt. Möglich ist jede Kombination der in der folgenden Tabelle aufgeführten Werte:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt wird.
MK_MBUTTON	Gesetzt, wenn die mittlere Maustaste gedrückt wird.
MK_RBUTTON	Gesetzt, wenn die rechte Maustaste gedrückt wird.
MK_SHIFT	Gesetzt, wenn die Shift-Taste gedrückt wird.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_LBUTTONUP

Beschreibung Diese Botschaft tritt auf, wenn der Anwender die linke Maustaste losläßt.

Parameter *wParam:* Enthält einen Wert, der den Status verschiedener virtueller Tasten anzeigt. Möglich ist jede Kombination der in der folgenden Tabelle aufgeführten Werte:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt wird.
MK_MBUTTON	Gesetzt, wenn die mittlere Maustaste gedrückt wird.
MK_RBUTTON	Gesetzt, wenn die rechte Maustaste gedrückt wird.
MK_SHIFT	Gesetzt, wenn die Shift-Taste gedrückt wird.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_MBUTTONDBLCLK

Beschreibung Diese Botschaft tritt auf, wenn der Anwender die mittlere Maustaste doppelklickt.

Parameter *wParam:* Enthält einen Wert, der den Status verschiedener virtueller Tasten anzeigt. Jede Kombination der folgenden Werte ist möglich:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt wird.
MK_LBUTTON	Gesetzt, wenn die linke Maustaste gedrückt wird.
MK_MBUTTON	Gesetzt, wenn die mittlere Maustaste gedrückt wird.
MK_RBUTTON	Gesetzt, wenn die rechte Maustaste gedrückt wird.
MK_SHIFT	Gesetzt, wenn die Shift-Taste gedrückt wird.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters. Nur Fenster, deren Fensterklasse den Stil *CS_DBLCLKS* hat, können Doppelklickbotschaften empfangen.

- Windows erzeugt eine Doppelklickbotschaft, wenn eine Maustaste gedrückt, losgelassen und dann innerhalb der vom System vorgegebenen Doppelklickzeit ein zweites Mal betätigt wird. Ein Doppelklick erzeugt vier Botschaften: Drücken der Taste, Loslassen der Taste, Doppelklickbotschaft und erneutes Loslassen der Taste.

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_MBUTTONDOWN

Beschreibung Diese Botschaft tritt auf, wenn der Anwender die mittlere Maustaste drückt.

Parameter *wParam:* Enthält einen Wert, der den Status verschiedener virtueller Tasten anzeigt. Jede Kombination der folgenden Werte ist möglich:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt wird.
MK_LBUTTON	Gesetzt, wenn die linke Maustaste gedrückt wird.
MK_RBUTTON	Gesetzt, wenn die rechte Maustaste gedrückt wird.
MK_SHIFT	Gesetzt, wenn die Shift-Taste gedrückt wird.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_MBUTTONUP

Beschreibung Diese Botschaft tritt auf, wenn der Anwender die mittlere Maustaste losläßt.

Parameter *wParam:* Enthält einen Wert, der den Status verschiedener virtueller Tasten anzeigt. Jede Kombination der folgenden Werte ist möglich:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt wird.
MK_LBUTTON	Gesetzt, wenn die linke Maustaste gedrückt wird.
MK_RBUTTON	Gesetzt, wenn die rechte Maustaste gedrückt wird.
MK_SHIFT	Gesetzt, wenn die Shift-Taste gedrückt wird.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_MDIACTIVATE

Beschreibung Eine Anwendung übergibt diese Botschaft an ein MDI-Client-Fenster (Multidokumentschnittstelle), um die Aktivierung eines untergeordneten MDI-Fensters zu veranlassen. Bei der Bearbeitung dieser Botschaft übergibt das Client-Fenster die Botschaft *WM_MDIACTIVATE* an das zu aktivierende und an das zu deaktivierende untergeordnete Fenster.

Parameter *wParam:* Wenn die Anwendung die Botschaft *WM_MDIACTIVATE* an das MDI-Client-Fenster übergibt, enthält der Parameter *wParam* das Fenster-Handle des zu aktivierenden untergeordneten MDI-Fensters. Übergibt das Client-Fenster die Botschaft an ein untergeordnetes Fenster, so hat der Parameter *wParam* einen Wert ungleich 0, wenn es aktiviert, 0, wenn es deaktiviert wird.

LOWORD(lParam): Das Handle des zu aktivierenden untergeordneten MDI-Fensters.

HIWORD(lParam): Das Handle des zu deaktivierenden untergeordneten MDI-Fensters.

Rückgabewert Nicht verwendet.

Hinweis Untergeordnete MDI-Fenster werden unabhängig vom MDI-Rahmenfenster aktiviert. Wird der Rahmen aktiv, erhält das zuletzt mit der Botschaft *WM_MDIACTIVATE* aktivierte, untergeordnete Fenster die Botschaft *WM_NCACTIVATE*, um Rahmen und Fenstertitel zu zeichnen. Weitere *WM_MDIACTIVATE*-Botschaften werden nicht an das Fenster übergeben.

Falls das zu deaktivierende untergeordnete MDI-Fenster auf

maximale Größe eingestellt ist, wird seine vorherige Größe wiederhergestellt und das zu aktivierende untergeordnete MDI-Fenster wird in seiner maximalen Größe dargestellt.

WM_MDICASCADE

Beschreibung Diese Botschaft überführt die einem MDI-Client-Fenster (Multidokumentschnittstelle) untergeordneten Fenster in das Format *Cascade* (überlappend), das heißt, er schichtet die untergeordneten Fenster leicht versetzt übereinander.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_MDICREATE

Beschreibung Diese Botschaft veranlaßt ein MDI-Client-Fenster (Multidokumentschnittstelle) zur Erstellung eines untergeordneten Fensters.

Parameter *wParam:* Nicht verwendet.

lParam: Enthält einen Zeiger auf eine Datenstruktur *MDICREATESTRUCT*.

Rückgabewert Der Rückgabewert enthält den Bezeichner des neuen Fensters im niederwertigen und 0 im höherwertigen WORD.

Hinweis Das Fenster wird mit den Stil-Bits *WS_CHILD, WS_CLIPSIBLINGS, WS_CLIPCHILDREN, WS_SYSMENU, WS_CAPTION, WS_THICKFRAME, WS_MINIMIZEBOX* und *WS_MAXIMIZEBOX* sowie den zusätzlichen Stil-Bits in der durch *lParam* bezeichneten *MDICREATESTRUCT*-Datenstruktur erstellt.

Windows fügt die Überschrift des neuen Fensters in das Fenstermenü des Rahmenfensters ein. Die Anwendung sollte alle dem Client-Fenster untergeordneten Fenster mit dieser Botschaft erzeugen.

Empfängt ein Client-Fenster eine Botschaft, welche die Aktivierung untergeordneter Fenster beeinflußt und ist das aktive Fenster vergrößert, stellt Windows das bisher aktivierte Fenster in alter Größe wieder her und vergrößert das neu aktivierte Fenster.

Bei Erzeugung des untergeordneten MDI-Fensters übergibt Windows die Botschaft *WM_CREATE* an das Fenster. Der Parameter *lParam* dieser Botschaft enthält einen Zeiger auf eine *CREATESTRUCT*-Datenstruktur. Das Feld *lpCreateParams* dieser Struktur enthält einen Zeiger auf die *MDICREATESTRUCT*-

Datenstruktur, die mit der das untergeordnete MDI-Fenster erzeugenden Botschaft *WM_MDICREATE* übergeben wird.

Eine Anwendung sollte keine zweite *WM_MDICREATE*-Botschaft übergeben, während die vorhergehende noch bearbeitet wird. Es sollte zum Beispiel keine *WM_MDICREATE*-Botschaft übergeben, während ein untergeordnetes MDI_FENSTER seine *WM_CREATE*-Botschaft abarbeitet.

WM_MDIDESTROY

Beschreibung Empfängt ein MDI-Client-Fenster (Multidokumentschnittstelle) diese Botschaft, wird ein untergeordnetes Fenster geschlossen.

Parameter *wParam:* Enthält das Fenster-Handle des untergeordneten Fensters.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft löscht die Überschrift des untergeordneten Fensters aus dem Rahmenfenster und schließt es dann.

Anwendungen sollten alle untergeordneten MDI-Fenster mit Hilfe dieser Botschaft schließen. Erhält ein Client-Fenster eine Botschaft, welche den Aktivierungsstatus eines untergeordneten Fensters verändert, und ist gleichzeitig das derzeit aktive untergeordnete MDI-Fenster vergrößert, wird es durch Windows auf normale Größe gebracht. Außerdem wird das neu aktivierte untergeordnete Fenster vergrößert.

WM_MDIGETACTIVE

Beschreibung Diese Botschaft liefert das aktive untergeordnete MDI-Fenster (Multidokumentschnittstelle) zusammen mit einem Flag, das anzeigt, ob das Fenster vergrößert ist.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert enthält im niederwertigen WORD das Handle des untergeordneten MDI-Fensters. Das höherwertige WORD enthält den Wert 1, wenn das Fenster vergrößert ist; andernfalls ist der Wert 0.

WM_MDIICONARRANGE

Beschreibung Diese Botschaft wird an ein MDI-Client-Fenster (Multidokumentschnittstelle) übergeben, um die verkleinerten unterge-

ordneten Dokumentenfenster anzuordnen. Es werden nur verkleinerte Fenster (Symbole) angeordnet.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_MDIMAXIMIZE

Beschreibung Diese Botschaft veranlaßt ein MDI-Client-Fenster (Multidokumentschnittstelle), ein untergeordnetes MDI-Fenster zu vergrößern. Windows paßt die Größe des zu öffnenden Fensters der des Client-Fensters an, wenn es zu groß ist. Das Systemmenü des untergeordneten Fensters wird in der Menüzeile des Rahmens positioniert und die Überschrift der Überschrift des Rahmenfensters hinzugefügt. Der Anwender kann so jederzeit das untergeordnete Fenster schließen oder wiederherstellen.

Parameter *wParam:* Enthält den Bezeichner des untergeordneten Fensters.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Erhält ein Client-Fenster eine Botschaft, welche den Aktivierungsstatus eines untergeordneten Fensters verändert, und ist gleichzeitig das derzeit aktive untergeordnete MDI-Fenster vergrößert, wird es durch Windows wiederhergestellt. Außerdem wird das neu aktivierte untergeordnete Fenster vergrößert.

WM_MDINEXT

Beschreibung Aktiviert das dem aktiven unmittelbar folgende untergeordnete MDI-Fenster (Multidokumentschnittstelle) und positioniert das noch aktive hinter allen anderen untergeordneten Fenstern.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Erhält ein Client-Fenster eine Botschaft, welche den Aktivierungsstatus eines untergeordneten Fensters verändert und ist gleichzeitig das derzeit aktive untergeordnete MDI-Fenster vergrößert, wird es durch Windows wiederhergestellt. Außerdem wird das neu aktivierte untergeordnete Fenster vergrößert.

WM_MDIRESTORE

Beschreibung Diese Botschaft stellt ein untergeordnetes MDI-Fenster (Multidokumentschnittstelle) aus dem Vollbild oder der Symboldarstellung wieder her.

Parameter *wParam:* Der Bezeichner des untergeordneten Fensters.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_MDISETMENU

Beschreibung Diese Botschaft tauscht das Menü eines MDI-Rahmenfensters (Multidokumentschnittstelle), das Pop-up-Menü des Fensters oder beide aus.

Parameter *wParam:* Nicht verwendet.

LOWORD(lParam): Das Handle eines neuen MDI-Rahmenfensters oder NULL.

HIWORD(lParam): Das Handle eines neuen Pop-up-Menüs oder NULL.

Rückgabewert Der Rückgabewert ist das Handle des Rahmenfenstermenüs, das durch die Botschaft ersetzt wird.

Hinweis Nach Übergabe dieser Botschaft muß die Anwendung die Funktion *DrawMenuBar* zur Aktualisierung der Menüleiste aufrufen. Tauscht die Botschaft ein Fenster-Pop-up-Menü aus, werden die Menüeinträge des untergeordneten MDI-Fensters aus dem vorhergehenden Fenstermenü gelöscht und zum neuen Pop-up-Menü hinzugefügt. Tauscht diese Botschaft ein MDI-Rahmenfenstermenü aus, während ein untergeordnetes MDI-Fenster vergrößert dargestellt wird, werden das Systemmenü und die wiederherzustellenden Steuerelemente aus dem vorhergehenden Rahmenfenstermenü entfernt und in das neue Menü eingefügt.

WM_MDITILE

Beschreibung Diese Botschaft veranlaßt ein MDI-Client-Fenster (Multidokumentschnittstelle) alle ihm untergeordneten Fenster im Format *Tile* (nebeneinander) darzustellen, das heißt, es ordnet alle untergeordneten Fenster so an, daß sie den Client-Bereich ausfüllen, ohne sich zu überlappen.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_MEASUREITEM

Beschreibung Diese Botschaft wird an den Besitzer eines besitzergezeichneten Schalters, Kombinationsfensters, Menüeintrages oder einer Liste übergeben, wenn das betreffende Element erzeugt wird.

Erhält der Besitzer diese Botschaft, füllt er vor der Antwort die durch den Parameter *lParam* bezeichnete *MEASUREITEMSTRUCT*-Datenstruktur aus.

Diese Datenstruktur informiert Windows über die Abmessungen des Steuerelementes. Werden die Typen *LBS_OWNERDRAWVARIABLE* oder *CBS_OWNERDRAWVARIABLE* zur Erstellung der Liste oder des Kombinationsfensters verwendet, empfängt der Besitzer eine Botschaft für jeden Eintrag im Steuerelement. In allen anderen Fällen wird die Botschaft nur einmal empfangen.

Parameter *wParam:* Nicht verwendet.

lParam: Enthält einen Zeiger auf eine *MEASUREITEMSTRUCT*-Datenstruktur.

Rückgabewert Nicht verwendet.

Hinweis Windows übergibt die Botschaft *WM_MEASUREITEM* vor Übergabe der *WM_INITDIALOG*-Botschaft an den Besitzer eines Kombinationsfensters oder einer Liste, wenn diese mit dem Stil-Typ *CBS_OWNERDRAWFIXED* oder *LBS_OWNERDRAWFIXED* erstellt wurden.

WM_MENUCHAR

Beschreibung Diese Botschaft wird übergeben, wenn der Anwender eine Tastenkombination betätigt, die keiner vordefinierten Tastenkombination des aktuellen Menüs entspricht. Die Botschaft wird an das Fenster übergeben, welches das Menü verwaltet.

Parameter *wParam:* Enthält den Wert des vom Anwender gedrückten ASCII-Zeichens.

LOWORD(lParam): MF_POPUP, wenn das Menü ein Pop-up-Menü ist, *MF_SYSMENU*, wenn es sich um ein Systemmenü handelt.

HIWORD(lParam): Das Handle des aktuellen Menüs.

Rückgabewert Das höherwertige WORD des Rückgabewerts enthält einen der folgenden Befehlscodes:

Code	Bedeutung
0	Weist Windows an, das vom Anwender gewählte Zeichen zu verwerfen und erzeugt einen Warnton.
1	Weist Windows an, das aktuelle Menü zu schließen.
2	Informiert Windows, daß im niederwertigen WORD des Rückgabewerts die Eintragsnummer eines bestimmten Eintrags abgelegt ist. Der betreffende Eintrag wird von Windows ausgewählt.

Hinweis

Das niederwertige WORD wird ignoriert, wenn das höherwertige den Wert 0 oder 1 hat. Werden Akzelerator-Tasten verwendet, um in einem Menü positionierte Bitmaps auszuwählen, sollte die Anwendung diese Botschaft bearbeiten.

WM_MENUSELECT

Beschreibung

Die Botschaft tritt nach Auswahl eines Menüeintrags durch den Anwender auf.

Parameter

wParam: Bezeichnet den ausgewählten Eintrag. Handelt es sich um einen Menüeintrag, enthält der Parameter *wParam* die zugehörige ID. Enthält der ausgewählte Eintrag ein Pop-up-Menü, so enthält *wParam* dessen Handle.

LOWORD(lParam): Entweder -1, oder eine Kombination der folgenden Werte:

Wert	Bedeutung
MF_BITMAP	Der Eintrag ist ein Bitmap.
MF_CHECKED	Der Eintrag ist markiert.
MF_DISABLED	Der Eintrag ist deaktiviert.
MF_GRAYED	Der Eintrag ist als nicht aktivierbar markiert.
MF_MOUSESELECT	Der Eintrag wurde mit der Maus ausgewählt.
MF_OWNERDRAW	Der Eintrag ist besitzergezeichnet.
MF_POPUP	Der Eintrag enthält ein Pop-up-Menü.
MF_SYSMENU	Der Eintrag ist im Systemmenü enthalten. Das höherwertige WORD enthält das Menü, auf das sich die Botschaft bezieht.

HIWORD(lParam): Ist Null, wenn das Systemmenü betroffen ist oder *LOWORD(lParam)* den Wert -1 hat. Andernfalls ist *HIWORD(lParam)* ein Handle auf das Menü.

Hinweis

Enthält *LOWORD(lParam)* den Wert -1 und *HIWORD(lParam)* den Wert 0, so wurde das Menü von Windows geschlossen, weil der Anwender die Esc-Taste gedrückt oder außerhalb des Menüs geklickt hat.

WM_MOUSEACTIVATE

Beschreibung Diese Botschaft tritt auf, wenn sich der Mauszeiger in einem nicht aktiven Fenster befindet und eine Maustaste gedrückt wird. Das übergeordnete Fenster erhält diese Botschaft nur, wenn sie vom untergeordneten Fenster an die Funktion *DefWindowProc* weitergegeben wird.

Parameter *wParam:* Enthält ein Handle des höchsten (dem zu aktivierenden Fenster übergeordneten) Fensters.

LOWORD(lParam): Eine der *HT*-Konstanten.

Diese Konstanten sind im Abschnitt »HT_XXXX« in Kapitel 4 »Windows-Stile und -Konstanten« beschrieben. Diese Werte werden auch von der Botschaft *WM_NCHITTEST* zurückgegeben.

HIWORD(lParam): Die Nummer der Mausbotschaft.

Rückgabewert Ist der Rückgabewert *MA_ACTIVATE*, wird das erste Fenster aktiviert, das die Botschaft erhält. Ist der Rückgabewert *MA_NOACTIVATE*, wird das erste Fenster, das die Botschaft erhält, nicht aktiviert. Ist der Rückgabewert *MA_ACTIVATEANDEAT*, wird das erste Fenster aktiviert, das die Botschaft erhält. Das Mausereignis wird verworfen. Andere Rückgabewerte sind nicht zulässig.

Hinweis Übergibt ein untergeordnetes Fenster die Botschaft an die Funktion *DefWindowProc*, wird sie von dieser Funktion vor jeder Bearbeitung an das übergeordnete Fenster weitergeleitet. Gibt das übergeordnete Fenster einen Wert ungleich 0 zurück, wird die Bearbeitung gestoppt.

WM_MOUSEMOVE

Beschreibung Diese Botschaft tritt auf, wenn der Anwender die Maus bewegt, während sich der Mauszeiger im Client-Bereich des Fensters befindet.

Parameter *wParam:* Enthält einen Wert, der den Zustand verschiedener virtueller Tasten anzeigt. Jede Kombination der folgenden Werte ist möglich:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt ist.
MK_LBUTTON	Gesetzt, wenn die linke Maustaste gedrückt ist.
MK_MBUTTON	Gesetzt, wenn die mittlere Maustaste gedrückt ist.
MK_RBUTTON	Gesetzt, wenn die rechte Maustaste gedrückt ist.
MK_SHIFT	Gesetzt, wenn die Shift-Taste gedrückt ist.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Querverweis *WM_LBUTTON-, WM_MBUTTON-, WM_RBUTTON-, WM_NCLBUTTON-, WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_MOVE

Beschreibung Diese Botschaft wird nach dem Verschieben eines Fensters übergeben.

Parameter *wParam:* Nicht verwendet.

LOWORD(lParam): Die neue x-Koordinate der oberen linken Ecke des Client-Bereichs des Fensters.

HIWORD(lParam): Die neue y-Koordinate der oberen linken Ecke des Client-Bereichs des Fensters.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten von Pop-up- und überlappenden Fenstern sind Bildschirmkoordinaten. Die Koordinaten von untergeordneten Fenstern sind relativ zum Client-Bereich des übergeordneten Fensters gebildet.

WM_NCACTIVATE

Beschreibung Diese Botschaft wird an ein Fenster übergeben, wenn dessen Nicht-Client-Bereich zum Anzeigen des aktiven bzw. inaktiven Status verändert werden muß.

Parameter *wParam:* Der Parameter *wParam* ist ungleich 0, wenn ein aktiver Titel bzw. ein aktives Symbol dargestellt werden soll. Bei inaktiven Titeln oder Symbolen ist er 0.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Die Funktion *DefWindowProc* zeichnet eine graue Titelleiste für inaktive Fenster und eine schwarze Titelleiste für aktive Fenster.

WM_NCCALCSIZE

Beschreibung Diese Botschaft wird übergeben, wenn die Größe des Client-Bereichs eines Fensters berechnet werden muß.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf eine *RECT-Daten*struktur, in der die Koordinaten des Fensterrechtecks (einschließlich Client-Bereich, Grenzlinien, Überschrift, Bildlaufleisten usw.) abgelegt sind.

Rückgabewert Nicht verwendet.

Hinweis Die Funktion *DefWindowProc* berechnet die Größe des Client-Bereichs, ausgehend von den Daten des Fensters (Existenz von Bildlaufleisten, Menüs, usw.). Das Ergebnis wird in der *RECT*-Datenstruktur abgelegt.

WM_NCCREATE

Beschreibung Diese Botschaft wird unmittelbar vor der Botschaft *WM_CREATE* beim Erzeugen eines Fensters übergeben.

Parameter *wParam:* Enthält das Handle des zu erstellenden Fensters.

lParam: Zeigt auf die *CREATESTRUCT*-Datenstruktur des Fensters.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn der Nicht-Client-Bereich erzeugt wird. Wenn ein Fehler auftritt, wird 0 zurückgegeben. Der Rückgabewert der Funktion *CreateWindow* ist dann ebenfalls Null.

Hinweis Bildlaufleisten werden initialisiert (Setzen von Position und Anzeigebereich der Bildlaufleiste) und der Fenstertext gesetzt. Intern verwendeter Speicher für Erzeugung und Verwaltung des Fensters wird reserviert. Diese Aufgaben übernimmt die Funktion *DefWindowProc*.

WM_NCDESTROY

Beschreibung Diese Botschaft teilt einem Fenster mit, daß sein Nicht-Client-Bereich freigegeben und entfernt wird. Die Funktion *DestroyWindow* übergibt die Botschaft *WM_NCDESTROY* im Anschluß an die *WM_DESTROY*-Botschaft an das Fenster. Diese Botschaft wird verwendet, um den dem Fenster zugewiesenen Speicherbereich freizugeben.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_NCHITTEST

Beschreibung Diese Botschaft wird an das Fenster übergeben, das den Mauszeiger enthält oder das die Funktion *SetCapture* verwendet hat, um die Mauseingabe auf sich zu ziehen. Die Botschaft wird bei jeder Mausbewegung übergeben.

Parameter *wParam:* Nicht verwendet.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Wird diese Botschaft an die Funktion *DefWindowProc* übergeben, ist der Rückgabewert eine der folgenden Konstanten:

Konstante	Bedeutung
HTBOTTOM	Auf der unteren horizontalen Grenzlinie des Fensters.
HTBOTTOMLEFT	Auf der unteren linken Ecke der Fenstergrenze.
HTBOTTOMRIGHT	Auf der unteren rechten Ecke der Fenstergrenze.
HTCAPTION	In einer Überschrift.
HTCLIENT	In einem Client-Bereich.
HTERROR	Wie *HTNOWHERE.* Zusätzlich erzeugt die Funktion DefWindowProc einen Warnlaut, um einen Fehler anzuzeigen.
HTGROWBOX	In einem Größenveränderungsfeld.
HTHSCROLL	In der horizontalen Bildlaufleiste.
HTLEFT	Auf der linken Grenzlinie des Fensters.
HTMENU	In einem Menübereich.
HTNOWHERE	Auf dem Bildschirmhintergrund oder einer Trennlinie zwischen zwei Fenstern.
HTREDUCE	In einem Verkleinerungsfeld.
HTRIGHT	Auf der rechten Grenzlinie des Fensters.
HTSIZE	Wie *HTGROWBOX.*
HTSYSMENU	In einem Menüfeld (Schließfeld in untergeordneten Fenstern).
HTTOP	Auf der oberen horizontalen Grenzlinie des Fensters.
HTTOPLEFT	Auf der oberen linken Ecke der Fenstergrenze.
HTTOPRIGHT	Auf der oberen rechten Ecke der Fenstergrenze.
HTTRANSPARENT	In einem Fenster, auf dem ein anderes Fenster liegt.
HTVSCROLL	In der vertikalen Bildlaufleiste.
HTZOOM	In einem Vergrößerungsfeld.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Normalerweise erhält das Fenster, in dem sich der Mauszeiger befindet, alle Botschaften. Dies kann mit der Funktion *SetCapture* umgangen werden.

Querverweis *WM_LBUTTON-, WM_MBUTTON-, WM_RBUTTON-, WM_NCLBUTTON-, WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCLBUTTONDBLCLK

Beschreibung Diese Botschaft wird an ein Fenster übergeben, wenn der Anwender mit der linken Maustaste innerhalb eines Nicht-Client-Bereichs des Fensters doppelklickt.

Parameter *wParam:* Enthält den Code, der von *WM_NCHITTEST* zurückgegeben wird. Weitere Informationen finden sich in der Beschreibung der Botschaft *WM_NCHITTEST*.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Windows erzeugt eine Doppelklickbotschaft, wenn eine Maustaste gedrückt, losgelassen und innerhalb der vom System vorgegebenen Doppelklickzeit ein zweites Mal betätigt wird. Ein Doppelklick erzeugt vier Botschaften: Drücken der Taste, Loslassen der Taste, Doppelklickbotschaft, erneutes Loslassen der Taste.

Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-, WM_MBUTTON-, WM_RBUTTON-, WM_NCLBUTTON-, WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCLBUTTONDOWN

Beschreibung Diese Botschaft wird an ein Fenster übergeben, wenn der Anwender die linke Maustaste drückt, während sich der Mauszeiger im Nicht-Client-Bereich des Fensters befindet.

Parameter *wParam:* Enthält den Code, der von *WM_NCHITTEST* zurückgegeben wird. Eine Beschreibung der Codes findet sich bei der Botschaft *WM_NCHITTEST*.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-, WM_MBUTTON-, WM_RBUTTON-, WM_NCLBUTTON-, WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCLBUTTONUP

Beschreibung Diese Botschaft wird an ein Fenster übergeben, wenn der Anwender die linke Maustaste losläßt, während sich der Mauszeiger im Nicht-Client-Bereich des Fensters befindet.

Parameter *wParam:* Enthält den Code, der von *WM_NCHITTEST* zurückgegeben wird. Eine Beschreibung des Codes findet sich bei der Botschaft *WM_NCHITTEST*.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters. Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-, WM_MBUTTON-, WM_RBUTTON-, WM_NCLBUTTON-, WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCMBUTTONDBLCLK

Beschreibung Diese Botschaft wird an ein Fenster übergeben, wenn der Anwender mit der mittleren Maustaste innerhalb des Nicht-Client-Bereichs des Fensters doppelklickt.

Parameter *wParam:* Enthält den Code, der von *WM_NCHITTEST* zurückgegeben wird. Eine Beschreibung des Codes findet sich bei der Botschaft *WM_NCHITTEST*.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Windows erzeugt eine Doppelklickbotschaft, wenn eine Maustaste gedrückt, losgelassen und dann innerhalb der vom System vorgegebenen Doppelklickzeit ein zweites Mal betätigt wird. Ein Doppelklick erzeugt vier Botschaften: Drücken der Taste, Loslassen der Taste, Doppelklickbotschaft, erneutes Loslassen der Taste.

Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCMBUTTONDOWN

Beschreibung Diese Botschaft wird an das Fenster übergeben, wenn der Anwender die mittlere Maustaste drückt, während sich der Mauszeiger innerhalb eines Nicht-Client-Bereichs des Fensters befindet.

Parameter *wParam:* Einer der Codes, die von *WM_NCHITTEST* zurückgegeben werden. Die Codes sind in der Botschaft *WM_NCHITTEST* beschrieben.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCMBUTTONUP

Beschreibung Diese Botschaft wird an das Fenster übergeben, wenn der Anwender die mittlere Maustaste losläßt, während sich der Mauszeiger innerhalb eines Nicht-Client-Bereichs des Fensters befindet.

Parameter *wParam:* Einer der Codes, die von *WM_NCHITTEST* zurückgegeben werden. Die Codes sind in der Botschaft *WM_NCHITTEST* beschrieben.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters. Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-, WM_MBUTTON-, WM_RBUTTON-, WM_NCLBUTTON-, WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCMOUSEMOVE

Beschreibung Diese Botschaft wird an ein Fenster übergeben, wenn der Mauszeiger innerhalb des Nicht-Client-Bereichs des Fensters bewegt wird.

Parameter *wParam:* Einer der Codes, die von *WM_NCHITTEST* zurückgegeben werden. Die Codes sind in der Botschaft *WM_NCHITTEST* beschrieben.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters

Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-, WM_MBUTTON-, WM_RBUTTON-, WM_NCLBUTTON-, WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCPAINT

Beschreibung Diese Botschaft wird übergeben, wenn der Fensterrahmen gezeichnet werden muß.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Die Funktion *DefWindowProc* zeichnet den Fensterrahmen. Anwendungen können diese Botschaft abfangen und individuelle Fensterrahmen malen. Der Clipping-Bereich eines Fensters ist immer rechteckig, auch wenn die Form des Rahmens hiervon abweicht.

WM_NCRBUTTONDBLCLK

Beschreibung Diese Botschaft wird an das Fenster übergeben, wenn der Anwender mit der rechten Maustaste doppelklickt, während sich der Mauszeiger innerhalb eines Nicht-Client-Bereichs des Fensters befindet.

Parameter *wParam:* Einer der Codes, die von *WM_NCHITTEST* zurückgegeben werden. Die Codes sind in der Botschaft *WM_NCHITTEST* beschrieben.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Windows erzeugt eine Doppelklickbotschaft, wenn eine Maustaste gedrückt, losgelassen und innerhalb der vom System vorgegebenen Doppelklickzeit ein zweites Mal betätigt wird. Ein Doppelklick erzeugt vier Botschaften: Drücken der Taste, Loslassen der Taste, Doppelklickbotschaft, erneutes Loslassen der Taste.

Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCRBUTTONDOWN

Beschreibung Diese Botschaft wird an das Fenster übergeben, wenn der Anwender die rechte Maustaste drückt, während sich der Mauszeiger innerhalb eines Nicht-Client-Bereichs des Fensters befindet.

Parameter *wParam:* Einer der Codes, die von *WM_NCHITTEST* zurückgegeben werden. Die Codes sind in der Botschaft *WM_NCHITTEST* beschrieben.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NCRBUTTONUP

Beschreibung Diese Botschaft wird an das Fenster übergeben, wenn der Anwender die rechte Maustaste losläßt, während sich der Mauszeiger innerhalb eines Nicht-Client-Bereichs des Fensters befindet.

Parameter *wParam:* Einer der Codes, die von *WM_NCHITTEST* zurückgegeben werden. Die Codes sind in der Botschaft *WM_NCHITTEST* beschrieben.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Die voreingestellte Aktion von *DefWindowProc* beinhaltet das Senden von geeigneten *WM_SYSCOMMAND*-Botschaften (in Abhängigkeit vom betroffenen Nicht-Client-Bereich).

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*

Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_NEXTDLGCTL

Beschreibung Diese Botschaft wird an die Fensterfunktion eines Dialogfensters geschickt, um den Fokus einem anderen Steuerelement zu übergeben. Das Resultat dieser Botschaft unterscheidet sich vom Resultat der Funktion *SetFocus* dadurch, daß die Botschaft *WM_NEXTDLGCTL* die Rahmenlinie um den Standardaktionsschalter modifiziert.

Parameter *wParam:* Ist *lParam* ungleich 0, bezeichnet *wParam* das Steuerelement, das den Fokus erhält. Ist *lParam* 0, stellt *wParam* ein Flag dar, das anzeigt, ob das nächste oder das vorhergehende Steuerelement den Fokus erhält. Ist *wParam* 0, erhält das nächste Steuerelement den Fokus, andernfalls das vorhergehende.

lParam: Enthält ein Flag, das anzeigt, wie Windows *wParam* interpretiert. Ist *lParam* ungleich 0, so stellt *wParam* ein Handle zum Steuerelement bereit, das den Fokus erhält. Andernfalls ist *wParam* ein Flag, das anzeigt, ob das nächste oder das vorhergehende Steuerelement den Fokus erhält.

Rückgabewert Nicht verwendet.

Hinweis Die Funktion *SendMessage* sollte diese Botschaft keinesfalls übergeben, wenn die Anwendung gleichzeitig andere Botschaften zum Setzen des Fokus bearbeitet. Stattdessen kann die Funktion *PostMessage* verwendet werden.

WM_PAINT

Beschreibung Diese Botschaft wird übergeben, wenn Windows oder eine Anwendung die Aktualisierung eines Teils eines Anwendungsfensters anfordert. Die Botschaft wird entweder beim Aufruf der Prozedur *UpdateWindow* oder durch die Funktion *DispatchMessage* übergeben, wenn die Anwendung die *WM_PAINT*-Botschaft durch Verwendung der Funktionen *GetMessage* oder *PeekMessage* erhält.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Querverweis *BeginPaint*, *EndPaint*

WM_PAINTCLIPBOARD

Beschreibung Diese Botschaft wird übergeben, wenn die Zwischenablage ein Daten-Handle im Format *CF_OWNERDISPLAY* enthält (d. h. der Besitzer der Zwischenablage sollte den Inhalt darstellen) und der Client-Bereich der Zwischenablage neu gezeichnet werden muß. Die Botschaft *WM_PAINTCLIPBOARD* wird an den Besitzer der Zwischenablage übergeben, um die Aktualisierung des Client-Bereichs (in Teilen oder komplett) der Zwischenablage anzufordern.

Parameter *wParam:* Enthält ein Handle zum Fenster der Zwischenablage.

LOWORD(lParam): Ein Speicher-Handle zur *PAINTSTRUCT*-Struktur, die den Teil der Zwischenablage definiert, der aktualisiert werden soll.

HIWORD(lParam): Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Um zu ermitteln, ob der gesamte Client-Bereich oder nur ein Teil aktualisiert werden muß, muß der Besitzer der Zwischenablage die Dimensionen des zu zeichnenden Bereichs im Feld *rcpaint* der *PAINTSTRUCT*-Datenstruktur mit den durch die Botschaft *WM_SIZECLIPBOARD* vorgegebenen vergleichen.

Anwendungen müssen die Funktion *GlobalLock* verwenden, um den Speicherbereich, der die *PAINTSTRUCT*-Datenstruktur enthält, zu sperren. Dieser Speicherbereich sollte mit Hilfe der Funktion *GlobalUnlock* vor Rück- oder Abgabe der Steuerung wieder freigegeben werden.

Daten und Format der Zwischenablage werden mit der Funktion *SetClipboardData* gesetzt.

WM_PAINTICON

Beschreibung Diese Botschaft wird an ein zum Symbol verkleinertes Fenster übergeben, wenn das Symbol gezeichnet werden soll. Die Botschaft kann nur von Fenstern empfangen werden, für die ein Klassensymbol vereinbart ist. Andernfalls wird die Botschaft *WM_PAINT* übergeben. Die Übergabe dieser Botschaft an die Funktion *DefWindowProc* ermöglicht es Windows, das Symbol mit Hilfe des Klassensymbols zu zeichnen.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_PALETTECHANGED

Beschreibung Diese Botschaft teilt allen Fenstern mit, daß das Fenster mit dem Fokus bei der Realisierung seiner virtuellen Farbskala die Systemfarbskala verändert hat. Dies erlaubt es Fenstern ohne Fokus, ihre virtuelle Farbskala zu realisieren und ihre Client-Bereiche zu aktualisieren.

Parameter *wParam:* Enthält das Handle des Fensters, das die Systemfarbskala verändert hat.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Um die Programmierung einer Endlosschleife zu vermeiden, sollte ein Fenster vor Realisierung der eigenen Farbskala sicherstellen, daß der Parameter *wParam* nicht sein eigenes Handle enthält.

WM_PARENTNOTIFY

Beschreibung Diese Botschaft wird an das übergeordnete Fenster übergeben, wenn das untergeordnete Fenster erstellt oder entfernt wird, oder wenn der Anwender eine Maustaste gedrückt hat, während der Mauszeiger über einem untergeordneten Fenster liegt. Wird das untergeordnete Fenster erstellt, übergibt Windows die Botschaft *WM_PARENTNOTIFY* unmittelbar vor Rückkehr der erzeugenden *CreateWindow*- bzw. *CreateWindowEx*-Funktion.

Soll das untergeordnete Fenster entfernt werden, übergibt Windows die Botschaft noch vor Entfernung des Fensters.

Parameter *wParam:* Bestimmt das dem übergeordneten Fenster mitzuteilende Ereignis. Möglich sind folgende Werte:

Wert	Bedeutung
WM_CREATE	Das untergeordnete Fenster wird gerade erzeugt.
WM_DESTROY	Das untergeordnete Fenster wird gerade entfernt.
WM_LBUTTONDOWN	Der Anwender hat ein untergeordnetes Fenster angeklickt.
WM_MBUTTONDOWN	Der Anwender hat ein untergeordnetes Fenster angeklickt.
WM_RBUTTONDOWN	Der Anwender hat ein untergeordnetes Fenster angeklickt.

LOWORD(lParam): Das Handle des untergeordneten Fensters, wenn *wParam WM_CREATE* oder *WM_DESTROY* ist. Andern-

falls enthält *LOWORD(lParam)* die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die ID des untergeordneten Fensters, wenn *wParam WM_CREATE* oder *WM_DESTROY* ist. Andernfalls enthält *HIWORD(lParam)* die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft wird an alle einem bestimmten Fenster übergeordneten Fenster bis hin zum Hauptfenster übergeben. Die Botschaft wird an die übergeordneten Fenster aller untergeordneten Fenster übergeben, außer das untergeordnete Fenster ist vom erweiterten Fensterstil *WS_EX_NOPARENTNOTIFY*. Die Funktion *CreateWindowEx* erzeugt Fenster mit erweiterten Fensterstilen. In der Standardeinstellung sind alle einem Dialogfenster untergeordneten Fenster vom Stil *WS_EX_NOPARENTNOTIFY*.

WM_PASTE

Beschreibung Diese Botschaft fügt die Daten aus der Zwischenablage in das Fenster an der aktuellen Mauszeigerposition ein. Die Daten werden nur dann eingefügt, wenn sie in der Zwischenablage im Format *CF_TEXT* vorliegen.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_QUERYDRAGICON

Beschreibung Diese Botschaft wird an ein zum Symbol verkleinertes Fenster übergeben, das vom Anwender verschoben wird und das kein definiertes Klassensymbol besitzt.

Verschiebt der Anwender das Symbol eines Fensters ohne Klassensymbol, ersetzt Windows dieses Symbol durch den Standardsymbolzeiger. Benötigt die Anwendung einen anderen Mauszeiger, der während des Verschiebens angezeigt werden soll, muß sie das Handle eines monochromen Mauszeigers zurückgeben. Der Mauszeiger muß zur Auflösung des Gerätetreibers passen.

Die Anwendung kann die Funktion *LoadCursor* aufrufen, um einen Mauszeiger aus den Ressourcen ihrer ausführbaren Datei zu laden und dessen Handle abzurufen.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert enthält im niederwertigen WORD das Handle des Mauszeigers, den Windows während des Verschiebens darstellen soll. Er ist 0, wenn Windows den Standardsymbolzeiger darstellen soll. Der Standardrückgabewert ist NULL.

WM_QUERYENDSESSION

Beschreibung Diese Botschaft wird erzeugt, wenn der Anwender den Befehl zum Beenden der Arbeitssitzung anwählt.

Gibt eine Anwendung 0 zurück, wird die Arbeitssitzung nicht beendet. In diesem Augenblick wird auch die weitere Übergabe von *WM_QUERYENDSESSION*-Botschaften durch Windows beendet und allen Anwendungen, die einen Wert ungleich 0 zurückgegeben haben, wird die Botschaft *WM_ENDSESSION* mit dem auf 0 gesetzten Parameter *wParam* übergeben.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn die Anwendung problemlos geschlossen werden kann. Andernfalls ist der Rückgabewert 0.

Hinweis Die Funktion *DefWindowProc* gibt standardmäßig einen Wert ungleich 0 zurück.

WM_QUERYNEWPALETTE

Beschreibung Diese Botschaft teilt einem Fenster mit, daß es den Fokus erhalten wird. Realisiert das Fenster beim Empfang des Fokus eine eigene virtuelle Farbskala, sollte es einen Wert ungleich 0, andernfalls 0 zurückgeben.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist ungleich 0, wenn das Fenster seine virtuelle Farbskala realisiert. Andernfalls wird 0 zurückgegeben.

WM_QUERYOPEN

Beschreibung Diese Botschaft wird an ein Symbol übergeben, wenn der Anwender dessen Öffnung zu einem Fenster anfordert.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Der Rückgabewert ist 0, wenn die Anwendung die Öffnung des Symbols verweigert und ungleich 0, wenn das Symbol geöffnet werden kann.

Hinweis Die Funktion *DefWindowProc* gibt einen Wert ungleich 0 zurück.

WM_QUIT

Beschreibung Diese Botschaft zeigt die Aufforderung zum Beenden einer Anwendung an. Sie wird erzeugt, wenn die Anwendung die Funktion *PostQuitMessage* aufruft. Die Funktion *GetMessage* gibt daraufhin 0 zurück.

Parameter *wParam:* Enthält den im Funktionsaufruf von *PostQuitMessage* vorgegebenen Exit-Code.

lParam: Nicht verwendet.

Rückgabewert Aufrufe der Funktion *GetMessage* geben 0 zurück, nachdem diese Botschaft übergeben wurde. Der Exit-Code in *wParam* muß gesichert und als Exit-Code der Anwendung verwendet werden.

WM_RBUTTONDBLCLK

Beschreibung Diese Botschaft tritt auf, wenn der Anwender mit der rechten Maustaste doppelklickt.

Parameter *wParam:* Enthält einen Wert, der den Status verschiedener virtueller Tasten anzeigt.

Jede Kombination der in der folgendenden Tabelle beschriebenen Werte ist möglich:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt wird.
MK_LBUTTON	Gesetzt, wenn die linke Maustaste gedrückt wird.
MK_MBUTTON	Gesetzt, wenn die mittlere Maustaste gedrückt wird.
MK_RBUTTON	Gesetzt, wenn die rechte Maustaste gedrückt wird.
MK_SHIFT	Gesetzt, wenn die Shift-Taste gedrückt wird.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Windows erzeugt eine Doppelklickbotschaft, wenn eine Maustaste gedrückt, losgelassen und innerhalb der vom System vorgegebenen Doppelklickzeit ein zweites Mal betätigt wird. Ein Doppelklick erzeugt vier Botschaften: Drücken der Taste, Loslassen der Taste, Doppelklickbotschaft, erneutes Loslassen der Taste.

Querverweis *WM_LBUTTON-, WM_MBUTTON-, WM_RBUTTON-, WM_NCLBUTTON-, WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_RBUTTONDOWN

Beschreibung Diese Botschaft tritt auf, wenn der Anwender die rechte Maustaste drückt.

Parameter *wParam:* Enthält einen Wert, der den Status verschiedener virtueller Tasten anzeigt. Jede Kombination der folgenden Werte ist möglich:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt wird.
MK_LBUTTON	Gesetzt, wenn die linke Maustaste gedrückt wird.
MK_MBUTTON	Gesetzt, wenn die mittlere Maustaste gedrückt wird.
MK-SHIFT	Gesetzt, wenn die Shift-Taste gedrückt wird.

LOWORD(lParam): Die x-Koordinate des Mauszeigers.

HIWORD(lParam): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Querverweis *WM_LBUTTON-, WM_MBUTTON-, WM_RBUTTON-, WM_NCLBUTTON-, WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_MOUSEMOVE* und *WM_NCMOUSEMOVE*.

WM_RBUTTONUP

Beschreibung Diese Botschaft tritt auf, wenn der Anwender die rechte Maustaste losläßt.

Parameter *wParam:* Enthält einen Wert, der den Status verschiedener virtueller Tasten anzeigt. Jede Kombination der folgenden Werte ist möglich:

Wert	Bedeutung
MK_CONTROL	Gesetzt, wenn die Ctrl-Taste gedrückt wird.
MK_LBUTTON	Gesetzt, wenn die linke Maustaste gedrückt wird.
MK_MBUTTON	Gesetzt, wenn die mittlere Maustaste gedrückt wird.
MK_SHIFT	Gesetzt, wenn die Shift-Taste gedrückt wird.

LOWORD(*lParam*): Die x-Koordinate des Mauszeigers.

HIWORD(*lParam*): Die y-Koordinate des Mauszeigers.

Rückgabewert Nicht verwendet.

Hinweis Die Koordinaten sind relativ zur oberen linken Ecke des Fensters.

Querverweis *WM_LBUTTON-*, *WM_MBUTTON-*, *WM_RBUTTON-*, *WM_NCLBUTTON-*, *WM_NCMBUTTON-* und *WM_NCRBUTTON-*Botschaften sowie *WM_NCMOUSEMOVE*.

WM_RENDERALLFORMATS

Beschreibung Diese Botschaft wird an eine zu entfernende Anwendung übergeben, wenn diese die Zwischenablage besitzt.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Die Anwendung sollte die Daten der Zwischenablage in alle Formate übersetzen, die sie erzeugen kann. Dabei muß ein Handle für jedes Format an die Funktion *SetClipboardData* übergeben werden. Dies stellt sicher, daß die Daten in der Zwischenablage auch nach Beendigung der Anwendung bearbeitet werden können.

WM_RENDERFORMAT

Beschreibung Durch diese Botschaft wird der Besitzer der Zwischenablage aufgefordert, die letzten in die Zwischenablage kopierten Daten im angegebenen Format zu formatieren. Anschließend wird das Handle der formatierten Daten an die Zwischenablage übergeben.

Parameter *wParam:* Bestimmt das Datenformat. Möglich sind die bei der Funktion *SetClipboardData* beschriebenen Formate.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_SETCURSOR

Beschreibung Diese Botschaft tritt dann auf, wenn kein Fenster den Mausfang besitzt, der Mauszeiger jeodch in einem Fenster bewegt wird.

Parameter *wParam:* Enthält das Handle des Fensters, in dem sich der Mauszeiger befindet.

LOWORD(lParam): Eine der *HT-Konstanten*, die in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »HTXXXX« beschrieben sind. Diese Werte werden auch von *WM_NCHITEST* zurückgegeben.

HIWORD(lParam): Die Nummer der Mausbotschaft.

Rückgabewert Wird 0 zurückgegeben, führt *DefWindowProc* seine Standardaktionen aus. Andernfalls finden keine weiteren Aktionen statt.

Hinweis Bevor Sie irgendetwas anderes tut, übergibt *DefWindowProc* die Botschaft *WM_SETCURSOR* an das übergeordnete Fenster. Gibt dieses Fenster einen Wert ungleich 0 zurück, wird die weitere Bearbeitung gestoppt.

Die Übergabe der Botschaft an ein übergeordnetes Fenster ermöglicht diesem die Kontrolle des Mauszeigers in einem untergeordneten Fenster.

Die Funktion *DefWindowProc* verwendet diese Botschaft außerdem, um dem Mauszeiger die Pfeilform zuzuweisen, wenn er sich in keinem Client-Bereich befindet. Andernfalls wird die registrierte klassenspezifische Form zugewiesen. hat der Parameter *LOWORD(lParam)* den Wert *HTERROR* und enthält *HIWORD(lParam)* eine Maustastendruck-Botschaft, so wird die Funktion *MessageBeep* aufgerufen. *HIWORD(lParam)* ist 0, wenn das Fenster in den Menümodus wechselt.

WM_SETFOCUS

Beschreibung Diese Botschaft wird übergeben, wenn ein Fenster den Fokus erhält.

Parameter *wParam:* Enthält das Handle des Fensters, das den Fokus abgibt (kann 0 sein).

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Um ein Caret darzustellen, sollte die Anwendung an dieser Stelle entsprechende Funktionen aufrufen.

WM_SETFONT

Beschreibung Diese Botschaft bestimmt die zur Textdarstellung verwendete Schrift eines Dialogfensters. Die Schrift sollte am besten gesetzt werden, wenn der Besitzer des Dialogfensters eine *WM_INITDIALOG*-Botschaft empfängt. Die Anwendung sollte nicht mehr benötigte Schriften mit der Funktion *DeleteObject* löschen, wenn zum Beispiel ein Steuerelement entfernt wird.

Diese Botschaft bewirkt keine Veränderung der Größe des Steuerelements. Um zu vermeiden, daß Windows Text einfügt, der nicht in die Umgrenzung eines Steuerelements paßt, sollte die Anwendung die Größe des Steuerfensters anpassen, bevor die Schrift geändert wird.

Parameter *wParam:* Enthält das Handle der Schrift. Hat dieser Parameter den Wert 0, stellt das Steuerelement den Text in der Standardsystemschrift dar.

lParam: Legt fest, ob das Steuerelement unmittelbar nach dem Setzen der Schrift neu gezeichnet werden soll. Ist *lParam* ungleich 0, wird das Steuerelement neu gezeichnet.

Rückgabewert Nicht verwendet.

Hinweis Bevor Windows ein Dialogfenster des Stils *DS_SETFONT* erzeugt, wird die Botschaft *WM_SETFONT* an das übergeordnete Fenster übergeben. Eine Anwendung erzeugt ein Dialogfenster des Stils *DS_SETFONT* durch Aufruf einer der folgenden Funktionen: *CreateDialogIndirect, CreateDialogIndirectParam, DialogBoxIndirect* oder *DialogBoxIndirectParam*

WM_SETREDRAW

Beschreibung Diese Botschaft wird von der Anwendung an ein Fenster übergeben, um die Darstellung von Veränderungen innerhalb des Fensters zuzulassen oder zu unterbinden.

Parameter *wParam:* Bestimmt den Status des Aktualisierungs-Flags. Ist der Parameter *wParam* ungleich 0, ist das Flag gesetzt.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft setzt oder löscht das Aktualisierungs-Flag. Dieses bezieht sich aber nicht auf Listen. Ist das Aktualisierungs-Flag gesetzt, kann ein Steuerelement unmittelbar nach jeder Veränderung aktualisiert werden, sonst nicht. Anwendungen, die einer Liste mehrere Einträge hinzufügen wollen, sollten das Flag vor dem Hinzufügen des letzten Eintrags setzen. In diesem Fall wird erst unmittelbar nach dem Hinzufügen des letzten Eintrags die Neudarstellung der Liste veranlaßt.

WM_SETTEXT

Beschreibung Diese Botschaft wird verwendet, um den Text eines Fensters zu setzen. Der zu setzende Text entspricht bei Editierfeldern und Editierfeldern in Kombinationsfenstern dem Inhalt der Felder. Bei Schaltern entspricht der Text dem Namen des Schalters. In allen anderen Fenstern ist der Text die Überschrift des Fensters.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf den nullterminierten String mit dem Fenstertext.

Rückgabewert Der Rückgabewert ist bei Listenfenstern *LB_ERRSPACE* bzw. *CB_ERRSPACE* bei Kombinationsfenstern, wenn der verfügbare Platz nicht ausreicht, um den Text in das Editierfeld zu setzen. Wird die Botschaft an ein Kombinationsfenster ohne Editierfeld übergeben, ist der Rückgabewert *CB_ERR*.

Hinweis Durch diese Botschaft wird die aktuelle Auswahl in der Liste oder im Kombinationsfenster nicht verändert. Mit Hilfe der Botschaft *CB_SELECTSTRING* kann eine Anwendung den Eintrag auswählen, der dem Text im Editierfeld entspricht.

WM_SHOWWINDOW

Beschreibung Diese Botschaft wird übergeben, wenn ein Fenster gezeigt oder verborgen wird: nach Aufruf der Funktion *ShowWindow*, nach Aktualisierung oder Vergrößerung eines überlappenden Fensters oder nach dem Schließen (Symbol) bzw. Öffnen (Bildschirmdarstellung) eines überlappenden bzw. eines Pop-up-Fensters. Wird ein überlappendes Fenster geschlossen, werden alle zugehörigen Pop-up-Fenster verborgen.

Parameter *wParam:* Bestimmt, ob ein Fenster gezeigt wird. Ist der Wert 0, wird das Fenster verborgen. Ist er ungleich 0, wird es gezeigt.

lParam: Bestimmt den Status des zu zeigenden Fensters. Der Parameter *lParam* hat den Wert 0, wenn die Botschaft aufgrund eines Aufrufs der Funktion *ShowWindow* übergeben wird.

Andernfalls sind folgende Werte möglich:

Wert	Bedeutung
SW_PARENTCLOSING	Es wird ein übergeordnetes Fenster geschlossen (Symbol) oder ein Pop-up-Fenster verborgen.
SW_PARENTOPENING	Es wird ein übergeordnetes Fenster geöffnet oder ein Pop-up-Fenster gezeigt.

Rückgabewert Nicht verwendet.

Hinweis Die Funktion *DefWindowProc* verbirgt oder zeigt das durch die Botschaft festgelegte Fenster.

WM_SIZE

Beschreibung Diese Botschaft wird nach Änderung der Größe eines Fensters übergeben.

Parameter *wParam:* Enthält einen Wert, der den Typ der gewünschten Größenänderung anzeigt. Möglich sind folgende Werte:

Wert	Bedeutung
SIZEFULLSCREEN	Fenster wurde zum Vollbild vergrößert.
SIZEICONIC	Fenster wurde zum Symbol verkleinert.
SIZENORMAL	Die Größe des Fensters wurde geändert, *SIZEICONIC* oder *SIZEFULLSCREEN* treffen jedoch nicht zu.
SIZEZOOMHIDE	Die Botschaft wird an alle Pop-up-Fenster übergeben, wenn ein anderes Fenster zum Vollbild vergrößert wurde.
SIZEZOOMSHOW	Die Botschaft wird an alle Pop-up-Fenster übergeben, wenn ein anderes Fenster in seiner voreingestellten Größe wiederhergestellt wurde.

Parameter *LOWORD(lParam):* Enthält die neue Breite des Client-Bereichs des Fensters.

HIWORD(lParam): Enthält die neue Höhe des Client-Bereichs des Fensters.

Rückgabewert Nicht verwendet.

Hinweis Wird als Ergebnis der Botschaft *WM_SIZE* eine der Funktionen *SetScrollPos* oder *MoveWindow* für ein untergeordnetes Fenster aufgerufen, sollte der Parameter *Redraw* bzw. *Repaint* ungleich 0 sein, um das Fenster zur Aktualisierung der Darstellung zu veranlassen.

WM_SIZECLIPBOARD

Beschreibung Diese Botschaft wird übergeben, wenn die Zwischenablage ein Daten-Handle des Formats *CF_OWNERDISPLAY* (d.h. der Besitzer der Zwischenablage sollte den Inhalt der Zwischenablage darstellen) enthält und die Größe des Fensters der Zwischenablage geändert wurde.

Parameter *wParam:* Das Handle des Fensters der Zwischenablage.

LOWORD(lParam): Eine *RECT*-Datenstruktur, die den Bereich des Fensters der Zwischenablage bestimmt, der neu dargestellt werden soll.

HIWORD(lParam): Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Die *WM_SIZECLIPBOARD*-Botschaft wird mit einem Rechteck mit den Koordinaten (0,0,0,0) als neuer Größe übergeben, wenn die Zwischenablage entfernt oder verkleinert werden soll. Dies erlaubt es dem Besitzer der Zwischenablage, die Bildschirmressourcen freizugeben.

Anwendungen müssen die Funktion *GlobalLock* verwenden, um den Speicher zu sperren, der die *PAINTSTRUCT*-Datenstruktur enthält. Sie sollten den Speicher mit Hilfe der Funktion *GlobalUnlock* wieder freigeben, bevor sie die Steuerung ab- oder zurückgeben.

WM_SPOOLERSTATUS

Beschreibung Diese Botschaft wird vom Druck-Manager übergeben, wenn ein Auftrag aus der Warteschlange des Druck-Managers entfernt oder in diese eingefügt wird.

Parameter *wParam:* Wird auf *PR_JOBSTATUS* gesetzt.

LOWORD(lParam): Anzahl der Druckaufträge, die sich aktuell in der Warteschlange des Druck-Managers befinden.

HIWORD(lParam): Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Diese Botschaft dient nur zu Informationszwecken.

WM_SYSCHAR

Beschreibung Diese Botschaft tritt auf, wenn *WM_SYSKEYUP*- und WM_*SYSKEYDOWN*-Botschaften übersetzt werden. Sie bestimmt den virtuellen Tastencode der Systemmenütaste.

Parameter *wParam:* Enthält den ASCII-Tastencode einer Systemmenütaste.

LOWORD(lParam): Wiederholungszähler. Der Wert ergibt sich aus der Zeit, die der Anwender die Taste gedrückt hält und entspricht der ermittelten Anzahl von Tastenwiederholungen.

HIWORD(lParam): Die Bits von *HIWORD(lParam)* können folgende Werte annehmen:

Bit	Bedeutung
0 bis 7	Enthalten den Scancode der Taste (OEM-abhängiger Wert).
8	Ist 1, wenn die gedrückte Taste eine zusätzliche Taste (z.B. eine Funktionstaste) ist.
9 bis 12	Werden entweder überhaupt nicht oder nur intern von Windows verwendet.
13	Ist 1, wenn gleichzeitig mit der Taste die Alt -Taste gedrückt wurde.
14	Ist 1, wenn die Taste vor Übergabe dieser Botschaft bereits gedrückt war.
15	Ist 1, wenn die Taste losgelassen, 0, wenn sie gedrückt wurde.

Rückgabewert Nicht verwendet.

Hinweis Ist der Kontextcode 0, kann die Botschaft an die Funktion *TranslateAccelerator* weitergegeben werden. Diese behandelt die Botschaft wie eine normale Tastaturbotschaft, nicht wie eine Systemtastenbotschaft. Dies ermöglicht es, Akzeleratortasten zusammen mit dem aktiven Fenster zu verwenden, auch wenn dieses nicht den Fokus hat.

Bei erweiterten IBM-Tastaturen mit 101 oder 102 Tasten gelten folgende Tasten als zusätzliche: die rechte Alt - und Ctrl - (bzw. Strg -)Taste im Hauptbereich der Tastatur, die Tasten Ins , Del , Home , End , PgUp , PgDown und die Cursor -Tasten im Block zwischen dem Nummernblock und der alphanumerischen Tastatur sowie die Divisions- und die Eingabetaste im Nummernblock. Auch andere Tastaturen unterstützen möglicherweise das Zusatztasten-Bit im Parameter *HIWORD (lParam)*.

WM_SYSCOLORCHANGE

Beschreibung Diese Botschaft bestimmt den Wechsel einer oder mehrerer Systemfarben. Windows übergibt die Botschaft an alle Hauptfenster, wenn eine Änderung in der Vorgabe der Systemfarben stattfindet.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Windows übergibt eine *WM_PAINT*-Botschaft an alle von der Systemfarbenänderung beeinflußten Fenster. Anwendungen, die Pinsel in den existierenden Systemfarben besitzen, sollten diese löschen und mit den neuen Systemfarben neu erstellen.

WM_SYSCOMMAND

Beschreibung Diese Botschaft wird übergeben, wenn der Anwender einen Befehl aus dem Systemmenü oder das Symbol für Vollbild- bzw. Symboldarstellung anwählt.

Parameter *wParam:* Bestimmt den Typ des angeforderten Systembefehls. Möglich ist einer der folgenden Werte:

Wert	Bedeutung
SC_CLOSE	Fenster schließen.
SC_HSCROLL	Horizontalen Bildlauf durchführen.
SC_KEYMENU	Das dem Tastendruck entsprechende Menü darstellen.
SC_MAXIMIZE, SC_ZOOM	Fenster als Vollbild darstellen.
SC_MINIMIZE, SC_ICON	Fenster als Symbol darstellen.
SC_MOUSEMENU	Das dem Mausklick entsprechende Menü darstellen.
SC_MOVE	Das Fenster verschieben.
SC_NEXTWINDOW	In das nächste Fenster bewegen.
SC_PREVWINDOW	In das vorhergehende Fenster bewegen.
SC_RESTORE	Stellt nach einer Maximierung oder Minimierung die ursprüngliche Größe des Fensters wieder her
SC_SIZE	Größe des Fensters einstellen.
SC_VSCROLL	Vertikalen Bildlauf durchführen.

LOWORD(lParam): x-Koordinate des Mauszeigers oder 0, wenn die Maus nicht verwendet wurde.

HIWORD(lParam): y-Koordinate des Mauszeigers oder 0, wenn die Maus nicht verwendet wurde.

Rückgabewert Nicht verwendet.

Hinweis Die Anfrage an das Systemmenü wird von der Funktion *DefWindowProc*, wie oben festgelegt, durchgeführt.

In *WM_SYSCOMMAND*-Botschaften werden die vier niederwertigen Bits des Parameters *wParam* intern von Windows verwendet. Um den korrekten Wert von *wParam* zu ermitteln, müssen Anwendungen den Wert des Parameters *wParam* mit 0xFFF0 verknüpfen, indem sie den bitweisen UND-Operator (&) verwenden.

Die Menüeinträge eines Systemmenüs können durch Verwendung der Funktionen *GetSystemMenu*, *AppendMenu*, *InsertMenu* und *ModifyMenu* modifiziert werden. Anwendungen, die das Systemmenü modifizieren, müssen die *WM_SYSCOMMAND*-Botschaften bearbeiten. Werden die Botschaften nicht bearbeitet, müssen sie an die Funktion *DefWindowProc* übergeben werden.

Durch die Anwendung hinzugefügte Befehlswerte können von dieser Funktion nicht verarbeitet werden, dies muß durch die Anwendung selbst geschehen. Anwendungen können jederzeit jeden Systembefehl ausführen, indem sie eine *WM_SYSCOMMAND*-Botschaft an die Funktion *DefWindowProc* übergeben. Akzelerator-Tasten, die für die Auswahl eines Eintrags im Systemmenü definiert wurden, werden in *WM_SYSCOMMAND*-Botschaften, alle anderen in *WM_COMMAND*-Botschaften übersetzt.

WM_SYSDEADCHAR

Beschreibung Diese Botschaft resultiert aus der Übersetzung einer *WM_SYSKEYUP*- und *WM_SYSKEYDOWN*-Botschaft. Sie bestimmt den Wert einer toten Taste.

Parameter *wParam:* Enthält den Zeichenwert einer toten Taste.

LOWORD(lParam): Anzahl der Tastenbetätigungen.

HIWORD(lParam): Anzahl der Tastenwiederholungen, die aus der Dauer des Tastendrucks ermittelt wird.

Rückgabewert Nicht verwendet.

Hinweis Tote Tasten enthalten das Umlaut-Zeichen und Akzente. Diese Botschaft kann verwendet werden, um auf Tastenbetätigungen zu reagieren, die selber nicht unbedingt zur Darstellung eines Zeichens führen (siehe auch die Beschreibungen der Funktionen *WM_CHAR*, *WM_DEADCHAR* und *WM_SYSCHAR*).

WM_SYSKEYDOWN

Beschreibung Dieser Botschaft wird übergeben, wenn der Anwender die Alt-Taste gedrückt hält und gleichzeitig eine andere Taste betätigt. Sie tritt auch auf, wenn aktuell kein Fenster den Fokus hat. In diesem Fall wird die Botschaft *WM_SYSKEYDOWN* an das aktive Fenster übergeben. Das empfangende Fenster kann zwischen den beiden Kontexten durch Überprüfung des Kontextcodes im Parameter *HIWORD(lParam)* unterscheiden.

Parameter *wParam:* Enthält den virtuellen Tastencode der gedrückten Taste.

LOWORD(lParam): Wiederholungszähler. Der Wert ergibt sich aus der Zeit, die der Anwender die Taste gedrückt hält und entspricht der ermittelten Anzahl von Tastenwiederholungen.

HIWORD(lParam): Die Bits von *HIWORD(lParam)* können folgende Werte annehmen:

Bit	Bedeutung
0 bis 7	Enthalten den Scancode der Taste (OEM-abhängiger Wert).
8	Ist 1, wenn die gedrückte Taste eine zusätzliche Taste (z.B. eine Funktionstaste) ist.
9 bis 12	Werden entweder überhaupt nicht oder nur intern von Windows verwendet.
13	Ist 1, wenn gleichzeitig mit der Taste die Alt-Taste gedrückt wurde.
14	Ist 1, wenn die Taste vor Übergabe dieser Botschaft bereits gedrückt war.
15	Ist 1, wenn die Taste losgelassen, 0, wenn sie gedrückt wurde.

Rückgabewert Nicht verwendet.

Hinweis Ist der Kontextcode 0, kann die Botschaft an die Funktion *TranslateAccelerator* weitergegeben werden. Diese behandelt die Botschaft wie eine normale Tastaturbotschaft, nicht wie eine Systemtastenbotschaft.

Dies ermöglicht es, Akzelerator-Tasten zusammen mit dem aktiven Fenster zu verwenden, auch wenn dieses nicht den Fokus hat.

Wegen der automatischen Tastenwiederholung kann es zu mehreren *WM_SYSKEYDOWN*-Botschaften kommen, bevor eine *WM_SYSKEYUP*-Botschaft übergeben wird. Der vorhergehende Tastenstatus kann verwendet werden, um festzustellen, ob die Botschaft *WM_SYSKEYDOWN* einen ersten oder einen wiederholten Tastendruck anzeigt.

Bei erweiterten IBM-Tastaturen mit 101 oder 102 Tasten gelten folgende Tasten als zusätzliche: die rechte Alt- und Ctrl-Taste im Hauptbereich der Tastatur, die Tasten Ins, Del, Home, End, PgUp, PgDown und die Cursor-Tasten im Block zwischen dem Nummernblock und der alphanumerischen Tastatur sowie die Divisions- und die Eingabetaste im Nummernblock.

Auch andere Tastaturen unterstützen möglicherweise das Zusatztasten-Bit im Parameter *HIWORD(lParam)*.

WM_SYSKEYUP

Beschreibung Diese Botschaft wird übergeben, wenn der Anwender eine Taste losläßt, die gleichzeitig mit der Alt-Taste gedrückt wurde. Sie tritt auch auf, wenn kein Fenster den aktuellen Fokus hat. In diesem Fall wird die Botschaft *WM_SYSKEYDOWN* an das aktive Fenster übergeben.

Das empfangende Fenster kann zwischen den beiden Kon-

texten durch Überprüfung des Kontextcodes im Parameter *HIWORD(lParam)* unterscheiden.

Parameter

wParam: Enthält den virtuellen Tastencode der losgelassenen Taste.

LOWORD(lParam): Wiederholungszähler. Der Wert ergibt sich aus der Zeit, die der Anwender die Taste gedrückt hält und entspricht der ermittelten Anzahl von Tastenwiederholungen.

HIWORD(lParam): Die Bits von *HIWORD(lParam)* können folgende Werte annehmen:

Bit	Bedeutung
0 bis 7	Enthalten den Scancode der Taste (OEM-abhängiger Wert).
8	Ist 1, wenn die gedrückte Taste eine zusätzliche Taste (zum Beispiel Funktionstaste oder Taste des numerischen Tastenblocks) ist.
9 bis 12	Werden entweder überhaupt nicht oder nur intern von Windows verwendet.
13	Ist 1, wenn gleichzeitig mit der Taste die Alt-Taste gedrückt wurde.
14	Ist 1, wenn die Taste vor Übergabe dieser Botschaft bereits gedrückt war.
15	Ist 1, wenn die Taste losgelassen, 0, wenn sie gedrückt wurde.

Rückgabewert

Nicht verwendet.

Hinweis

Ist der Kontextcode 0, kann die Botschaft an die Funktion *TranslateAccelerator* weitergegeben werden. Diese behandelt die Botschaft wie eine normale Tastaturbotschaft, nicht wie eine Systemtastenbotschaft. Dies ermöglicht es, Akzeleratortasten zusammen mit dem aktiven Fenster zu verwenden, auch wenn dieses nicht den Fokus hat.

Bei erweiterten IBM-Tastaturen mit 101 oder 102 Tasten gelten folgende Tasten als zusätzliche: die rechte Alt- und Ctrl-Taste im Hauptbereich der Tastatur, die Tasten Ins, Del, Home, End, PgUp, PgDown und die Cursor-Tasten im Block zwischen dem Nummernblock und der alphanumerischen Tastatur sowie die Divisions- und die Eingabetaste im Nummernblock. Auch andere Tastaturen unterstützen möglicherweise das Zusatztasten-Bit im Parameter *HIWORD(lParam)*.

Bei anderen erweiterten Tastaturen mit 102 Tasten wird die rechte Alt-Taste wie eine Ctrl-Alt-Tastenkombination behandelt.

Die folgende Tabelle zeigt die Folge der Botschaften, die sich aus dem Drücken und Loslassen dieser Taste ergeben:

Reihenfolge	Botschaft	Virtueller Tastencode (*HIWORD(lParam)*)
1	*WM_KEYDOWN*	*VK_CONTROL*
2	*WM_KEYDOWN*	*VK_MENU*
3	*WM_KEYUP*	*VK_CONTROL*
4	*WM_SYSKEYUP*	*VK_MENU*

WM_TIMECHANGE

Beschreibung Diese Botschaft tritt auf, wenn eine Anwendung die Systemzeit verändert. Alle Anwendungen, die dies tun, sollten diese Botschaft an alle Hauptfenster übergeben.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Die Anwendung kann die Funktion *SendMessage* verwenden, um *WM_TIMECHANGE* an alle Hauptfenster zu übergeben. Dazu muß der Parameter *Wnd* auf 0xFFFF gesetzt sein.

WM_TIMER

Beschreibung Diese Botschaft tritt auf, wenn die Zeitbegrenzung des vorgegebenen Timers überschritten wurde.

Parameter *wParam:* Enthält die Timer-ID, ein Integer, der den Timer bezeichnet.

lParam: Zeigt auf eine Funktion, die bei der Erstellung des Timers an die Funktion *SetTimer* übergeben wurde. Windows ruft die festgelegte Funktion direkt auf, wenn der Parameter *lParam* den Wert 0 hat. Andernfalls wird die Botschaft *WM_TIMER* an die Fensterfunktion übergeben.

Rückgabewert Nicht verwendet.

WM_UNDO

Beschreibung Diese Botschaft macht die vorhergehende Operation rückgängig. Nach Übergabe der Botschaft an ein Editierfeld wird der gerade gelöschte Text wiederhergestellt bzw. der zuletzt hinzugefügte Text gelöscht.

Parameter *wParam:* Nicht verwendet.

lParam: Nicht verwendet.

Rückgabewert Nicht verwendet.

WM_VKEYTOITEM

Beschreibung Diese Botschaft wird als Reaktion auf eine *WM_KEYDOWN*-Botschaft von einer Liste des Stils *LBS_WANTKEYBOARDINPUT* an seinen Besitzer übergeben.

Parameter *wParam:* Enthält den virtuellen Tastencode der vom Anwender gedrückten Taste.

LOWORD(lParam): Das Handle der Liste.

HIWORD(lParam): Die aktuelle Position des Caret.

Rückgabewert Der Rückgabewert bestimmt die Aktion, die von der Anwendung als Reaktion auf diese Botschaft ausgeführt wurde.

Ein Rückgabewert von -2 zeigt an, daß die Anwendung alle Aspekte der Auswahl eines Eintrags bearbeitet hat und keine weiteren Aktionen in der Liste gewünscht werden. Der Rückgabewert -1 zeigt an, daß die Liste die Standardaktion als Reaktion auf den Tastendruck durchführen soll. Ist der Rückgabewert 0 oder größer, bestimmt er den Index des Eintrags in der Liste und zeigt an, daß die Liste die Standardaktion durchführen soll, die dem Tastendruck und dem vorgegebenen Eintrag entspricht.

WM_VSCROLL

Beschreibung Diese Botschaft wird übergeben, wenn der Anwender die vertikale Bildlaufleiste anklickt.

Parameter *wParam:* Enthält einen Bildlaufleistencode, der die Bildlaufwünsche des Anwenders bestimmt. Folgende Werte sind möglich:

Wert	Bedeutung
SB_BOTTOM	Bildlauf nach unten.
SB_ENDSCROLL	Ende des Bildlaufs.
SB_LINEDOWN	Bildlauf eine Zeile nach unten.
SB_LINEUP	Bildlauf eine Zeile nach oben.
SB_PAGEDOWN	Bildlauf eine Seite nach unten.
SB_PAGEUP	Bildlauf eine Seite nach oben.
SB_THUMBPOSITION	Bildlauf zu einer absoluten Position.
SB_THUMBTRACK	Mauszug zur festgelegten Position.
SB_TOP	Bildlauf nach oben.

LOWORD(lParam): Nicht verwendet.

HIWORD(lParam): Enthält das Handle der Bildlaufleiste. *HIWORD(lParam)* ist 0, wenn sie mit dem Stil *WS_VSCROLL* erzeugt wurde.

Rückgabewert Nicht verwendet.

Hinweis Die Botschaft *SB_THUMBTRACK* wird üblicherweise von Anwendungen benutzt, die während des Bewegens der Positionsmarke reagieren (z.B. durch gleichzeitiges Scrollen des Fensterinhalts).

Führt eine Anwendung einen Bildlauf in einem Fenster durch, muß sie die Positionsmarke durch die Funktion *SetScrollPos* zurücksetzen.

WM_VSCROLLCLIPBOARD

Beschreibung Diese Botschaft wird übergeben, wenn die Zwischenablage ein Daten-Handle des Formats *CF_OWNERDISPLAY* (d.h. der Besitzer der Zwischenablage sollte den Inhalt der Zwischenablage darstellen) enthält und ein Ereignis in der vertikalen Bildlaufleiste der Zwischenablage auftritt.

Parameter *wParam:* Enthält das Handle des Fensters der Zwischenablage.

LOWORD(lParam): Enthält einen der folgenden Bildlaufleistencodes:

Wert	Bedeutung
SB_BOTTOM	Bildlauf nach unten rechts.
SB_ENDSCROLL	Ende des Bildlaufs.
SB_LINEDOWN	Bildlauf eine Zeile nach unten.
SB_LINEUP	Bildlauf eine Zeile nach oben.
SB_PAGEDOWN	Bildlauf eine Seite nach unten.
SB_PAGEUP	Bildlauf eine Seite nach oben.
SB_THUMBPOSITION	Bildlauf zu einer absoluten Position.
SB_TOP	Bildlauf nach oben links.

HIWORD(lParam): Nicht verwendet.

Rückgabewert Nicht verwendet.

Hinweis Der Besitzer der Zwischenablage sollte die Funktion *InvalidateRect* verwenden oder die Zwischenablage wie gewünscht neu darstellen. Die Positionsmarke sollte mit Hilfe der Funktion *SetScrollPos* zurückgesetzt werden. Daten und Format der Zwischenablage werden mit *SetClipboardData* gesetzt.

WM_WININICHANGE

Beschreibung Diese Botschaft wird übergeben, wenn die Initialisierungsdatei WIN.INI verändert wird. Ändert eine Anwendung die Datei WIN.INI, sollte sie diese Botschaft an alle Hauptfenster übergeben.

Parameter *wParam:* Nicht verwendet.

lParam: Zeigt auf einen String, der den Namen des geänderten Abschnitts angibt. Die eckigen Klammern sind in diesen String nicht eingeschlossen.

Rückgabewert Nicht verwendet.

Hinweis Die Anwendung kann die Funktion *SendMessage* verwenden, um die Botschaft an alle Hauptfenster zu übergeben. Dabei muß der Parameter *Wnd* auf 0xFFFF gesetzt sein.

Auch wenn es eigentlich falsch ist, übergeben einige Anwendungen diese Botschaft mit dem auf NULL gesetzten Parameter *lParam*. Empfängt eine Anwendung eine derartige Botschaft, sollte sie solche Abschnitte der Datei WIN.INI überprüfen, die sie selbst betreffen könnten.

3. Windows-Typen

Windows definiert eine Anzahl von Datentypen, Handle und andere, meist komplexere Strukturen. Dieses Kapitel beschreibt diese Typen in alphabetischer Reihenfolge zusammen mit der jeweiligen Header-Datei.

Windows definiert auch die meisten entsprechenden Zeiger, die hier allerdings nicht aufgeführt werden. So sind zum Beispiel *PBOOL* und *LPBOOL* als **near-** bw. **far**-Zeiger auf BOOL-Werte definiert.

Tabelle 3.1 Eine Übersicht nach Typen:

Typ	Name
DATA	*BOOL*
	BYTE
	DWORD
	LPSTR
	WORD
HANDLE	*GLOBALHANDLE*
	HBITMAP
	HBRUSH
	HCURSOR
	HDC
	HFONT
	HICON
	HMENU
	HPALETTE
	HPEN
	HRGN
	HSTR
	HWND
DDE	*DDEACK*
	DDEADVISE
	DDEDATA
	DDEPOKE
MISCELLANEOUS	Alle anderen Typen

ATOM-Typ **windows.h**

Definition
```
typedef WORD ATOM;
```

Beschreibung *ATOM* definiert einen 16-Bit-Zahlentyp, der ein Atom oder eine Botschaft bezeichnet, die zwischen DDE betreibenden Anwendungen ausgetauscht werden soll.

BITMAP-Typ **windows.h**

Definition
```
typedef struct tagBITMAP
  {
    int bmType;
    int bmWidth;
    int bmHeight;
    int bmWidthBytes;
    BYTE bmPlanes;
    BYTE bmBitsPixel;
    LPSTR bmBits;
  } BITMAP;
```

Beschreibung *BITMAP*-Strukturen werden von den Funktionen *CreateBitmapIndirect* und *GetObject* benutzt, um die Größe, die Farben und die Bitwerte für ein Bitmap zu beschreiben.

Das Element *bmType* definiert den Bitmap-Typ. Der Wert Null kennzeichnet ein virtuelles Bitmap.*bmWidth* und *bmHeight* definieren die Breite des Bitmap in Pixel bzw. die Höhe in Rasterzeilen. Beide müssen größer als Null sein. *bmWidthBytes* bestimmt die Anzahl von Bytes in jeder Rasterzeile und muß eine gerade Zahl sein.

bmPlanes und *bmBitsPixel* bestimmen die Anzahl der Farbebenen bzw. die Anzahl der benachbarten Farbbits auf jeder Ebene.

bmBits ist ein far-Zeiger auf die eigentlichen Bits, die das Bitmap bilden. Die Bits haben die Form eines Arrays von Bytes.

Querverweis *CreateBitmapIndirect, GetObject*

BITMAPCOREHEADER-Typ **windows.h**

Definition
```
typedef struct tagBITMAPCOREHEADER
  {
    DWORD bcSize;
    WORD bcWidth;
    WORD bcHeight;
    WORD bcPlanes;
    WORD bcBitCount;
  } BITMAPCOREHEADER;
```

Beschreibung *BITMAPCOREHEADER* definiert die Größe und die Farben eines geräteunabhängigen Bitmap. Die *BITMAPCOREHEADER*-Strukturen werden als Teil von *BITMAPCOREINFO*-Strukturen verwendet, um geräteunabhängige Bitmaps vollständig zu definieren.

bcSize ist die Anzahl der Bytes in der Struktur *BITMAPCOREHEADER*.

bcWidth und *bcHeight* sind die Breite bzw. die Höhe des Bitmap in Pixel. *bcPlanes* bestimmt die Anzahl von Ebenen für das Zielgerät und muß auf 1 gesetzt werden. *bcBitCount* bestimmt die Anzahl von Bits pro Pixel. Zulässige Werte für *bcBitCount* sind 1, 4, 8 und 24.

Die Werte von *bcBitCount* haben folgende Bedeutung:

- Wenn *bcBitCount* 1 ist, ist das Bitmap monochrom, die Farbtabelle muß zwei Einträge enthalten und jedes Bit im Bitmap steht für ein Pixel. Ein nicht gesetztes Bit (0) repräsentiert die erste Farbe in der Tabelle, ein gesetztes Bit (1) die zweite.

- Wenn *bcBitCount* 4 ist, hat das Bitmap bis zu 16 Farben, die von 0 bis 15 numeriert sind, so daß jedes Pixel im Bitmap vier Bits zur Kennzeichnung seiner Farbe beansprucht. Die Farbtabelle enthält 16 Einträge. Folglich repräsentiert jedes Byte im Bitmap zwei Pixel, das erste im höherwertigen und das zweite im niederwertigen Halb-Byte.

- Wenn *bcBitCount* 8 ist, hat das Bitmap bis zu 256 Farben, so daß ein ganzes Byte benötigt wird, um jedes Pixel zu repräsentieren. Jedes Byte im Bitmap steht damit für einen Index in die Farbtabelle zwischen 0 und 255.

- Wenn *bcBitCount* 24 ist, hat das Bitmap bis zu 2^{24} Farben. Es gibt keine Farbtabelle und jedes Pixel wird von drei Bytes repräsentiert, die die Intensitäten des roten, grünen und gelben Farbanteils im Pixel bestimmen.

BITMAPCOREINFO-Typ

windows.h

Definition

```
typedef struct tagBITMAPCOREINFO
    {
    BITMAPCOREHEADER bmciHeader;
    RGBTRIPLE bmciColors[1];
  } BITMAPCOREINFO;
```

Beschreibung *BITMAPCOREINFO*-Strukturen kombinieren Information über die Größe und die Farbe aus einer *BITMAPCOREHEADER*-Struktur mit einer Farbinformationstabelle, um ein geräteunabhängiges Bitmap vollständig zu definieren.

bmciHeader ist eine *BITMAPCOREHEADER*-Struktur, die die

Größen- und Farbinformationen für das Bitmap enthält. Das Element *bcSize* von *bmciHeader* kann verwendet werden, um einen Offset für *bmciColors* anzugeben.

bmciColors ist ein offenes Array von *RGBTRIPLE*-Strukturen. Die Dimensionierung auf eins geschieht hier der korrekten Syntax wegen, die wirkliche Anzahl von Elementen in diesem Array wird vom Element *bcBitCount* in *bmciHeader* bestimmt.

bmciColors kann entweder ein Array von RGB-Farbstrukturen sein oder ein Array von Indizes in die aktuell realisierte virtuelle Palette. Die Art, in der das Feld interpretiert wird, hängt vom Parameter *Usage* ab, der an die Funktion übergeben wird, die auf das geräteunabhängige Bitmap zugreift. In Kapitel 4 »Windows-Stile und -Konstanten« finden sich im Abschnitt »DIB_XXXX« weitere Informationen.

Der Einsatz von RGB-Werten für *bmciColors* wird sehr empfohlen, wenn das Bitmap nicht ausschließlich von einer einzigen Anwendung verwendet wird. Wenn das Bitmap in einer Datei gespeichert wird oder an ein anderes Programm übertragen wird, sollte keine Palettenindizierung verwendet werden.

BITMAPFILEHEADER-Typ

windows.h

Definition

```
typedef struct tagBITMAPFILEHEADER
  {
    WORD bfType;
    DWORD bfSize;
    WORD bfReserved1;
    WORD bfReserved2;
    DWORD bfOffBits;
  } BITMAPFILEHEADER;
```

Beschreibung

BITMAPFILEHEADER-Strukturen definieren den Header einer geräteunabhängigen Bitmap-Datei. Sie enthalten Daten, die die Typgröße und das Layout der Bitmap-Datei definieren.

Eine geräteunabhängige Bitmap-Datei besteht aus einer Struktur *BITMAPFILEHEADER*, gefolgt von einer der beiden Strukturen *BITMAPINFO* oder *BITMAPCOREINFO* und anschließend den eigentlichen Bitmap-Daten. Die Elemente des Bitmap-Datei-Header sind wie folgt definiert:

- *bfType* bestimmt den Typ der Datei. Er muß BM sein.
- *bfSize* bestimmt die Größe der Datei in 4-Byte-Blöcken (DWORDs).

bfReserved1 und *bfReserved2* sind von Windows reserviert.

bfOffBits bestimmt, nach wievielen Bytes ab Dateianfang die eigentliche Bitmap-Information beginnt.

BITMAPINFO-Typ

windows.h

Definition

```
typedef struct tagBITMAPINFO
  {
    BITMAPINFOHEADER bmiHeader;
    RGBQUAD bmiColors[1];
  } BITMAPINFO;
```

Beschreibung

BITMAPINFO-Strukturen enthalten Größen- und Farbinformationen für geräteunabhängige Bitmaps für Windows 3.0. Das eigentliche Bitmap ist als ein offenes Array von Bytes definiert, das die Pixel des Bitmaps repräsentiert. *bmiHeader* ist eine *BITMAPINFOHEADER*-Struktur, die die Größen- und Farbformatierung für das Bitmap enthält. *bmiColors* ist ein Array von *RGBQUAD*-Strukturen, die die Farben des Bitmaps definieren. Die Anzahl der Einträge im Array wird vom Wert des Elementes *biBitCount* der durch *bmiHeader* bezeichneten Struktur bestimmt.

BITMAPINFOHEADER-Typ

windows.h

Definition

```
typedef struct tagBITMAPINFOHEADER
  {
    DWORD biSize;
    DWORD biWidth;
    DWORD biHeight;
    WORD biPlanes;
    WORD biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    DWORD biXPelsPerMeter;
    DWORD biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
  } BITMAPINFOHEADER;
```

Beschreibung

BITMAPINFOHEADER-Strukturen werden von *BITMAPINFO*-Strukturen verwendet, um die Größen- und Farbformatierung eines geräteunabhängigen Bitmaps für Windows 3.0 zu definieren.

biSize bestimmt die Größe der Struktur in Bytes. *biWidth* und *biHeight* bestimmen jeweils Breite und Höhe des Bitmaps in Pixel. *biPlanes* ist die Anzahl der Ebenen für das Zielgerät und muß auf 1 gesetzt sein. *biBitCount* bestimmt die Anzahl der Bits, die erforderlich sind, um jedes Pixel im Bitmap zu beschreiben. *biCompression* gibt den Typ der Kompression an, der für das Bitmap verwendet werden soll; es kann eine der *BI_*-Konstanten sein, die in Kapitel 4 »Windows-Stile und -Konstanten« im Abschnitt »BI_XXXX« beschrieben sind.

biSizeImage ist die Größe des Bitmap-Bildes in Byte. *biXPelsPerMeter* und *biYPelsPerMeter* bestimmen die horizontale bzw. die vertikale Auflösung des Zielgerätes für das Bitmap.

biClrUsed wird verwendet, um die Anzahl der Farbtabelleneinträge zu bestimmen, die vom Bitmap tatsächlich benutzt werden. Der Wert *biBitCount* bestimmt die Maximalzahl der Einträge: Eine 0 in *biClrUsed* zeigt an, daß die Maximalzahl verwendet wurde, ein Wert zwischen 1 und 23 zeigt an, auf wieviele Farben tatsächlich zugegriffen wurde. Wenn *biBitCount* 24 ist, ist *biClrUsed* die Größe der Referenzfarbtabelle, die von Windows verwendet wurde, um die Leistungsfähigkeit der Farbpalette zu optimieren.

biClrImportant bestimmt die Anzahl der Farben, die für die Darstellung des Bitmaps als wichtig erachtet wurden.

Ein Wert von 0 zeigt an, daß alle Farben wichtig sind. Die Bedeutungen der Werte von *biBitCount* entsprechen exakt denen, die für das Element *bcBitCount* der *BITCOREHEADER*-Struktur beschrieben sind.

BOOL-Typ

windows.h

Definition

```
typedef int BOOL;
```

Beschreibung Ein 16-Bit-Boolean-Wert. 0 entspricht FALSE, ein Wert ungleich 0 dem logischen »wahr«. TRUE ist 1.

BYTE-Typ

windows.h

Definition

```
typedef unsigned char BYTE;
```

Beschreibung Ein vorzeichenloser 8-Bit-Zahlentyp. Er kann die Werte 0 bis 255 aufnehmen.

CATCHBUF-Typ

windows.h

Definition

```
typedef int CATCHBUF[9];
```

Beschreibung *CATCHBUF* ist eine Struktur, die den Status aller Systemregister enthält. Sie wird von den Funktionen *Catch* und *Throw* verwendet.

CLIENTCREATESTRUCT-Typ

windows.h

Definition

```
typedef struct tagCLIENTCREATESTRUCT
  {
    HANDLE hWindowMenu;
    WORD idFirstChild;
  } CLIENTCREATESTRUCT;
```

Beschreibung *CLIENTCREATESTRUCT* wird beim Erzeugen von MDI-Client-Fenster verwendet und nimmt Fenster-ID und Menü-Daten auf. Das Feld *hWindowMenu* ist ein Handle zum Menü der Anwendung. *idFirstChild* ist die Fenster-ID des ersten untergeordneten Fensters der MDI-Anwendung. Die IDs untergeordneter Fenster werden von Windows eindeutig zugewiesen und verwaltet.

COLORREF-Typ

windows.h

Definition

```
typedef DWORD COLORREF;
```

Beschreibung *COLORREF* ist ein 32-Bit-Zahlentyp, dessen Werte Farben entsprechen. Er wird von zahlreichen GDI-Funktionen verwendet. Er kann, abhängig vom Wert des höherwertigen Bytes im höherwertigen WORD, auf drei verschiedene Arten interpretiert werden.

- Wenn das höchstwertige Byte 0 ist, stehen die nächsten drei Bytes für RGB-Farbintensitäten (jeweils für Blau, Grün und Rot). Damit repräsentiert der Wert 0x00FF0000 reines Blau mit voller Intensität, 0x0000FF00 reines Grün, 0x000000FF reines Rot, 0x00000000 Schwarz und 0x00FFFFFF Weiß. RGB-Werte können am einfachsten mit der Funktion *RGB* in *COLORREF*-Werte konvertiert werden.

- Wenn das höchstwertige Byte 1 ist, muß das nächste Byte 0 sein. Das niederwertige WORD (die nächsten beiden Bytes) bilden einen Index in eine virtuelle Palette. Somit ist 0x01000000 der Index 0 (der erste Eintrag) einer Palette. Palettenindizes vom Typ Integer können mit dem Makro *PALETTEINDEX* in *COLORREF*-Werte konvertiert werden.

- Wenn das höchstwertige Byte 2 ist, repräsentieren die nächsten drei Bytes RGB-Farbintensitäten (wie mit RGB-Werten, die für ein höchstwertiges Byte von 0 beschrieben wurden). Der Wert wird jedoch hier an die ähnlichste Farbe in der virtuellen Palette im aktuellen Gerätekontext angeglichen. Palettenbezogene RGB-*COLORREF*-Werte können mit dem Makro *PaletteRGB* aus RGB-Werten ermittelt werden.

Um zu erreichen, daß die palettenindex- oder palettenbezogenen *COLORREF*-Werte mit einem Gerätekontext arbeiten, muß eine Anwendung mit eigener Palette mit *SelectPalette* ihre Palette in den Gerätekontext auswählen, für den sie geschrieben ist und danach mit *RealizePalette* realisieren. Damit verfügt sie über Zeichenfunktionen, die die richtigen Farben aus der Palette verwenden. Analog dazu muß die Palette vor dem Erzeugen eines virtuellen Zeichenwerkzeugs ausgewählt und realisiert werden, damit die Farben gültig werden.

COMPAREITEMSTRUCT-Typ **windows.h**

Definition

```
typedef struct tagCOMPAREITEMSTRUCT
  {
    WORD CtlType;
    WORD CtlID;
    HWND hwndItem;
    WORD itemID1;
    DWORD itemData1;
    WORD itemID2;
    DWORD itemData2;
  } COMPAREITEMSTRUCT;
```

Beschreibung *COMPAREITEMSTRUCT*-Strukturen werden dazu verwendet, Einträge in sortierten, besitzergezeichneten Kombinationsfenstern oder Listen zu vergleichen. Das Hinzufügen von Einträgen zu solchen Steuerelementen erzeugt eine Botschaft *WM_COMPAREITEM*, wobei einer der Parameter dieser Botschaft ein Zeiger auf eine *COMPAREITEMSTRUCT*-Struktur ist. Nach dem Empfang dieser Botschaft vergleicht der Besitzer die Einträge in der Struktur und gibt ergebnisabhängige Werte zurück.

Siehe auch die Beschreibung der *WM_COMPAREITEM*-Botschaft in Kapitel 2 »Windows-Botschaften«.

CtlType enthält eine *ODT_*-Konstante, die anzeigt, daß das Steuerelement ein besitzergezeichnetes Kombinationsfenster (*ODT_COMBOBOX*) oder Liste (*ODT_LISTBOX*) ist. *CtlID* und *hwndItem* sind die Steuerelement-ID bzw. das Fenster-Handle für das Steuerelement.

itemID1 und *itemData1* bestimmt den Index des ersten Eintrags in der Liste oder im Kombinationsfenster, der verglichen wird, sowie die zugehörigen Daten für diesen Eintrag. Die Daten von *itemData1* entsprechen denen des Parameters *lParam* der Botschaft, die den Eintrag zur Liste hinzufügt. *ItemID2* und *itemData2* bieten entsprechende Informationen für den zweiten Eintrag.

COMSTAT-Typ **windows.h**

Definition

```
typedef struct tagCOMSTAT
  {
    BYTE fCtsHold: 1;
    BYTE fDsrHold: 1;
    BYTE fRlsdHold: 1;
    BYTE fXoffHold: 1;
    BYTE fXoffSent: 1;
    BYTE fEof: 1;
    BYTE fTxim: 1;
```

```
  WORD cbInQue;
  WORD cbOutQue;
} COMSTAT;
```

Beschreibung *COMSTAT*-Strukturen beinhalten Statusdaten einer Schnittstelle und werden von der Funktion *GetCommError* verwendet.

Die *COM_*-Konstanten geben Auskunft darüber, warum die Übertragung wartet: *fCtsHold* (für Clear-to-send), *fDsrHold* (für Data-set-ready), *fRlsdHold* (für Receive-line-signal-detect), *fXoffHold* (weil ein *XOFF*-Zeichen empfangen wurde), *fXoffSent* (weil ein *XOFF*-Zeichen gesendet wurde und der Empfänger *XON* erwartet), *fEof* (*EOF*-Zeichen empfangen) *fTxim* (Zeichen wartet auf Übertragung). Die Felder *cbInQue* und *cbOutQue* bestimmen die Anzahl der Zeichen in der Empfangs- bzw. Sendewarteschlange.

Querverweis *GetCommError*

CREATESTRUCT-Typ

windows.h

Definition

```
typedef struct tagCREATESTRUCT
  {
    LPSTR lpCreateParams;
    HANDLE hInstance;
    HANDLE hMenu;
    HWND hwndParent;
    int cy;
    int cx;
    int y;
    int x;
    LONG style;
    LPSTR lpszName;
    LPSTR lpszClass;
    DWORD dwExStyle;
  } CREATESTRUCT;
```

Beschreibung *CREATESTRUCT*-Strukturen werden verwendet, um Initialisierungsparameter an Fensterfunktionen einer Anwendung zu übergeben. Das Feld *lpCreateParams* zeigt auf Erstellungsdaten des Fensters. *hInstance*, *hMenu* und *hwndParent* sind Handles zur Modulinstanz des zu dem Fenster gehörigen Moduls, zum Menü für das Fenster bzw. zum übergeordneten Fenster des neuen Fensters. *hwndParent* ist 0, wenn das erzeugte Fenster das Hauptfenster der Anwendung ist.

cx und *cy* bestimmen die Breite und die Höhe des Fensters, *x* und *y* die horizontalen und vertikalen Koordinaten der oberen linken Ecke des Fensters. Die Koordinaten sind relativ zu einem übergeordneten Fenster, wenn ein solches vorhanden ist.

style enthält die Stil-Flags für das Fenster, *lpszName* und *lpszClass* zeigen auf die nullterminierten Strings, die den Namen bzw. Klassennamen des Fensters enthalten. *dwExStyle* enthält erweiterte Stil-Informationen für das Fenster.

DCB-Typ

windows.h

Definition

```
typedef struct tagDCB
  {
    BYTE Id;
    WORD BaudRate;
    BYTE ByteSize;
    BYTE Parity;
    BYTE StopBits;
    WORD RlsTimeout;
    WORD CtsTimeout;
    WORD DsrTimeout;

    BYTE fBinary: 1;
    BYTE fRtsDisable:1;
    BYTE fParity: 1;
    BYTE fOutxCtsFlow:1;
    BYTE fOutxDsrFlow:1;
    BYTE fDummy: 2;
    BYTE fDtrDisable:1;

    BYTE fOutX: 1;
    BYTE fInX: 1;
    BYTE fPeChar: 1;
    BYTE fNull: 1;
    BYTE fChEvt: 1;
    BYTE fDtrflow: 1;
    BYTE fRtsflow: 1;
    BYTE fDummy2: 1;

    char XonChar;
    char XoffChar;
    WORD XonLim;
    WORD XoffLim;
    char PeChar;
    char EofChar;
    char EvtChar;
    WORD TxDelay;

    } DCB;
```

Beschreibung

DCB-Strukturen enthalten Steuerelementinformationen für diejenigen Schnittstellen, die von den Funktionen *BuildCommDCB, GetCommState* und *SetCommState* verwendet werden.

Das Feld *Id* ist die ID der Schnittstelle. Wenn das höherwer-

tige Bit gesetzt ist (vgl. die Maske *LPTx*), handelt es sich um einen parallelen, andernfalls um einen seriellen Anschluß.

BaudRate, *ByteSize*, *Parity* und *StopBits* definieren die Übertragungsparameter der Schnittstelle. *ByteSize* ist die Anzahl der Bits in jedem Zeichen, der Wert kann zwischen 4 und 8 (einschließlich) liegen. Für *Parity* wird eine der Datenübertragungskonstanten *EVENPARITY*, *MARKPARITY*, *NOPARITY*, *ODDPARITY* oder *SPACEPARITY* eingesetzt. *StopBits* ist eine der Datenübertragungskonstanten *ONESTOPBIT*, *ONE5STOPBITS*, oder *TWOSTOPBITS*. Die Datenübertragungskonstanten werden in Kapitel 4 »Windows-Stile und -Konstanten« beschrieben.

RlsTimeout, *CtsTimeout* und *DsrTimeout* bestimmen die maximale Zeit, die die Schnittstelle auf die Signale *RLSD*, *CTS* bzw. *DSR* warten soll, in Millisekunden.

fBinary bestimmt, ob der Anschluß im Binärmodus arbeitet. Im Binärmodus fließen alle Daten unverändert durch, im Nicht-Binärmodus wird das Zeichen *EOF* (*EofChar*) verwendet, um das Datenende zu kennzeichnen.

Jedes Bit der folgenden Tabelle repräsentiert einen Ein- / Aus-Schalter für bestimmte Fehlerprüfungen. Auf einzelne Bits kann mittels der *DCB_*-Konstanten zugegriffen werden.

Bit	Bedeutung, wenn gesetzt
fRtsDisable	*RTS* gesperrt.
fParity	Paritätsprüfung eingeschaltet.
fOutxCtsFlow	*CTS* beim Senden überwacht.
fOutxDsrFlow	*DSR* beim Senden überwacht.
fDummy	Reserviert.
fDtrDisable	*DTR* gesperrt.
fOutX	*XON/XOFF*-Steuerung beim Senden verwendet.
fInX	*XON/XOFF*-Steuerung beim Empfangen verwendet.
fPeChar	Paritätsfehler werden ersetzt.
fNull	Nullzeichen werden verworfen.
fChEvt	Das Zeichen *EvtChar* wird als Ereignis gekennzeichnet.
fDtrFlow	*DTR* wird beim Empfangen als Flußsteuerung verwendet.
fRtsFlow	*RTS* wird beim Empfangen als Flußsteuerung verwendet.
fDummy2	Reserviert.

XonChar und *XoffChar* bestimmen jeweils die Werte der Zeichen *XON* und *XOFF* für Senden bzw. Empfangen. *XonLim* ist die Anzahl der Zeichen in der Empfangswarteschlange, die das Senden eines *XON*, *XoffLim* die Anzahl der Zeichen, die das Senden eines *XOFF* auslöst.

PeChar, EofChar und *EvtChar* definieren die Zeichen, die jeweils verwendet werden, um Paritätsfehler zu ersetzen, das Ende der Daten zu signalisieren und ein Ereignis anzuzeigen.

TxDelay wird gegenwärtig nicht verwendet.

Querverweis *BildCommDCB, GetCommState, SetCommState*

DDEACK-Typ DDE.H

Definition

```
typedef struct
  {
    unsigned bAppReturnCode: 8,
             reserved: 6,
             fBusy: 1,
             fAck: 1;
  } DDEACK;
```

Beschreibung *DDEACK*-Strukturen enthalten Quittierungsinformationen, die in einem Parameter einer Botschaft *WM_DDE_ACK* als Antwort auf jede DDE-Botschaft – ausgenommen *WM_DDE_-INITIATE* – gesendet wurden.

Auf *bAppReturnCode* kann mit Hilfe der Maske DDE_APPRETURNCODE zugegriffen werden. Er enthält einen anwendungsspezifischen Rückgabecode. Die folgenden sechs Bits sind für Windows reserviert. Ist das Bit *fBusy* gesetzt, zeigt es an, daß die Anwendung nicht auf die Anforderung reagieren kann. Dies gilt allerdings nur, wenn *fAck* Null ist. Ist das Bit *fAck* gesetzt, wurde die Anforderung akzeptiert.

DDEADVISE-Typ DDE.H

Definition

```
typedef struct
  {
    unsigned reserved: 14,
             fDeferUpd: 1,
             fAckReq: 1;
    int cfFormat;
  } DDEADVISE;
```

Beschreibung Die Struktur *DDEADVISE* enthält eine Anforderung an einen DDE-Server und wird in einem Parameter einer Botschaft *WM_DDE_ADVISE* übergeben.

Der Zugriff auf *fAckReq* und *fDeferUpd* kann mit Hilfe der *DDE_*-Konstanten durchgeführt werden. Ist das Bit *fAckReq* gesetzt, bittet es den *DDE*-Server, seine *WM_DDE_DATA*-Botschaften mit gesetztem *fAckReq*-Bit zu senden. Damit kann eine Kontrolle des Botschaftsaustausches implementiert werden. Das gesetzte Bit *fDeferUpd* entspricht der Anforderung an den Server, seine *WM_DDE_DATA*-Botschaften mit einem

Null-Handle (*hData*) zu übergeben, um so dem Client mitzuteilen, daß sich die Daten geändert haben. Der Client kann dann eine *WM_DDE_REQUEST*-Botschaft übergeben, um die aktualisierten Daten abzufragen.

cfFormat bestimmt das bevorzugte Datenformat des Client durch Verwendung der *CF_*-Zwischenablageformatkonstanten.

DDEDATA-Typ DDE.H

Definition

```
typedef struct
  {
    unsigned unused: 12,
             fResponse: 1,
             fRelease: 1,
             reserved: 1,
             fAckReq: 1;
    int cfFormat;
    BYTE Value[1];
  } DDEDATA;
```

Beschreibung *DDEDATA*-Strukturen enthalten Daten, die von einer Anwendung zur anderen übertragen werden. Sie werden als Parameter in den Botschaften *WM_DDE_DATA* übergeben.

Auf *fResponse*, *fRelease* und *fAckReq* kann mit Hilfe der *DDE_*-Konstanten zugegriffen werden.

Ist *fResponse* gesetzt, werden die verknüpften Daten als Reaktion auf eine Anforderung des Client übergeben. Das gesetzte Bit *fRelease* enthält die Aufforderung an den Client, die Daten nach deren Bearbeitung freizugeben. Das Bit *reserved* ist für Windows reserviert. Das gesetzte *fAckReq*-Bit zeigt an, daß der Client eine Empfangsbestätigung in der Form einer *WM_DDE_ACK*-Botschaft senden sollte.

cfFormat enthält eine *CF_*-Zwischenablageformatkonstante, die das Format anzeigt, in dem die Daten an den Client gesendet werden. *Value* enthält die Daten, die übertragen werden, in dem von *cfFormat* vorgegebenen Format.

DDEPOKE-Typ DDE.H

Definition

```
typedef struct
  { unsigned unused: 13,
             fRelease: 1,
             fReserved: 2;
    int cfFormat;
    BYTE Value[1];
  } DDEPOKE;
```

Beschreibung *DDEPOKE*-Strukturen enthalten vom Empfänger nicht angeforderte Daten und begleiten eine Botschaft *WM_DDE_POKE*.

Das gesetzte Bit *fRelease* zeigt an, daß der Empfänger die Daten nach deren Bearbeitung freigeben sollte. Mit Hilfe der *DDE_*-Konstanten kann auf *fRelease* zugegriffen werden. Zwei Bits (*fReserved*) sind reserviert.

cfFormat bezeichnet das bevorzugte Datenformat des Client, wobei eine der *CF_*-Zwischenablageformatkonstanten verwendet wird.

Value enthält die Daten, die übertragen werden, in dem von *cfFormat* vorgegebenen Format.

DELETEITEMSTRUCT-Typ **windows.h**

Definition
```
typedef struct tagDELETEITEMSTRUCT
  {
    WORD CtlType;
    WORD CtlID;
    WORD itemID;
    HWND hwndItem;
    DWORD itemData;
  } DELETEITEMSTRUCT;
```

Beschreibung *DELETEITEMSTRUCT*-Strukturen werden verwendet, um einen aus einem besitzergezeichneten Kombinations- oder Listenfenster zu löschenden Eintrag zu beschreiben. Die Botschaft *WM_DELETEITEM* gelangt zum Besitzer des Fensters, wobei der Parameter *lParam* auf eine *DELETEITEMSTRUCT*-Struktur zeigt.

CtlType kennzeichnet den Typ des Steuerelements: Liste (*ODT_LISTBOX*) oder Kombinationsfenster (*ODT_COMBOBOX*). *CtlId* ist die Steuerelement-ID des Fensters, *itemID* der Index des gelöschten Eintrags.

hwndItem ist das Fenster-Handle des Steuerelements und *itemData* der 32-Bit-Wert des indizierten Eintrags.

DEVMODE-Typ **drivinit.h**

Definition
```
typedef STRUCT_DEVICEMODE
{
  char dmDeviceName[CCHDEVICENAME];
  WORD dmSpecVersion;
  WORD dmDriverVersion;
  WORD dmSize;
  WORD dmDriverExtra;
  DWORD dmFields;
  short dmOrientation;
  short dmPaperSize;
```

```
  short dmPaperLength;
  short dmPaperWidth;
  short dmScale;
  short dmCopies;
  short dmDefaultSource;
  short dmPrintQuality;
  short dmColor;
  short dmDuplex;
} DEVMODE;
```

Beschreibung

DevMode-Strukturen werden von den Funktionen *DeviceCapabilities* und *ExtDeviceMode* verwendet, um Informationen über einen Druckertreiber aufzunehmen.

Das 32-Byte-Array *dmDeviceName* enthält einen nullterminierten String, der die Bezeichnung des unterstützten Gerätes angibt. *dmSpecVersion* ist die Versionsnummer der Datenspezifikation, gegenwärtig 0x0300. *dmDriverVersion* ist die Versionsnummer des Treibers, die vom Entwickler vergeben wird. *dmSize* ist die Größe der Struktur ohne Berücksichtigung des Elements *dmDriverExtra*.

dmFields ist ein 32-Bit-Wert, der anzeigt, welche der verbleibenden Elemente initialisiert wurden. Jedes Bit entspricht einem Element:

Flag	Feld
DM_COLOR	*dmColor*
DM_COPIES	*dmCopies*
DM_DEFAULTSOURCE	*dmDefaultSource*
DM_DUPLEX	*dmDuplex*
DM_ORIENTATION	*dmOrientation*
DM_PAPERLENGTH	*dmPaperLength*
DM_PAPERSIZE	*dmPaperSize*
DM_PAPERWIDTH	*dmPaperWidth*
DM_PRINTQUALITY	*dmPrintQuality*
DM_SCALE	*dmScale*
DM_SPECVERSION	*dmSpecVersion*

Das Feld *dmOrientation* wählt die Papierausrichtung – Portrait oder Landscape (Hoch- bzw. Querformat) – unter Verwendung der Konstanten *DMORIENT_* aus.

Das Feld *dmPaperSize* wählt die Papiergröße aus, wobei eine der *DMPAPER_*-Konstanten verwendet wird. Mit den Feldern *dmPaperLength* und *dmPaperWidth* kann die im Feld *dmPaperSize* angegebene Papierlänge und -breite überschrieben werden.

Das Feld *dmScale* skaliert die gedruckte Ausgabe um einen Faktor von *dmScale*/100. Beispielsweise reduziert ein Wert von 75 Bilder auf 75% ihrer normalen Größe.

Das Feld *dmCopies* bestimmt die Anzahl der zu druckenden Kopien.

Das Feld *dmDefaultSource* kennzeichnet mit einer der *DMBIN_*-Konstanten den Schacht, aus dem das Papier per Voreinstellung eingezogen wird.

dmPrintQuality bestimmt die zu verwendende Druckauflösung. Es kann entweder eine der geräteunabhängigen *DMRES_*-Konstanten (negative Werte) angegeben werden oder eine geräteabhängige positive Zahl, deren Wert die Punkte pro Zoll bestimmt.

Das Feld *dmColor* wählt zwischen Farb- und Monochromdruck, wobei die *DMCOLOR_*-Konstanten eingesetzt werden. *dmDuplex* wählt mittels der *DMDUP_*-Konstanten ein- oder zweiseitigen Druck.

DLGITEMTEMPLATE **windows.h**

Beschreibung *DLGITEMTEMPLATE* ist keine echte Struktur. Sie kann weder in C++noch in einer anderen Programmiersprache dargestellt werden, da nullterminierte String-Daten in sie eingebettet sind. Die Eintragsvorlage eines Dialogfensters sollte deshalb als Strukturvorlage, nicht als Struktur angesehen werden.

DLGITEMTEMPLATE enthält fünf Integer, gefolgt von einem long-Integer und zwei nullterminierten Strings. Anschließend folgt noch ein Byte und möglicherweise zusätzliche Daten. Die ersten vier Integer (*dtilX*, *dtilY*, *dtilCX* und *dtilCY*) beschreiben Position und Abmessung eines Steuerelements. Sie verhalten sich relativ zum Ursprung des Dialogfensters. Die Einheiten entsprechen denen, die bei *DLGTEMPLATEHEADER* beschrieben wurden.

Es folgt der Integer *dtilID*, der die Eintrags-ID des Steuerelements bestimmt.

Der *long*-Integer *dtilStyle* enthält den Stil des Dialogfenstereintrags.

Nach der Stilinformation folgt ein nullterminierter String, der den Namen der Fensterklasse enthält. Mögliche Werte sind *BUTTON*, *EDIT*, *STATIC*, *LISTBOX*, *SCROLLBAR* und *COMBOBOX*.

Dem Klassennamen folgt ein weiterer nullterminierter String, *dtilText*, der den Text des Eintrags enthält.

Hinter den Strings kommt ein *dtilInfo* genanntes Byte, das die Anzahl von Bytes der nachfolgenden zusätzlichen Informationen enthält. Der Wert Null zeigt an, daß keine zusätzlichen Informationen existieren. In diesem Fall wird *DLGITEMTEMPLATE* durch dieses Byte beendet. Ist *dtilInfo* ungleich Null, enthält die nächste Bytegruppe *dtilData* Informationen, die im Element *CreateParams* einer *CREATESTRUCT*-Struktur an die Funktion *CreateWindow* übergeben werden.

Die Windows-Funktionen *CreateDialogIndirect*, *CreateDialogIndirectParam*, *DialogBoxIndirect* und *DialogBoxIndirectParam* übernehmen Zeiger auf eine Windows-Datenstruktur namens *DLGTEMPLATE*.

DLGTEMPLATE

windows.h

Beschreibung

DLGTEMPLATE ist keine echte Struktur. Sie kann weder in C++ noch in einer anderen Programmiersprache dargestellt werden, da in ihr ein nullterminierter String eingebettet ist. Eine Dialogfenstervorlage sollte deshalb nicht als Struktur, sondern als Strukturvorlage betrachtet werden.

Eine Dialogfenstervorlage besteht aus drei Teilen. Nur der erste Teil wird benötigt. Am Anfang steht ein *DLGTEMPLATEHEADER*. Enthält *dtStyle* in diesem Header die Konstante *DS_SETFONT*, wird für das Dialogfenster eine andere als die Systemschrift verwendet. Darauf müssen unmittelbar zwei Einträge folgen, die die Schrift beschreiben: ein vorzeichenbehafteter 8-Bit-Wert, der die Punkt-Größe der Schrift bestimmt und anschließend ein nullterminierter String, der den Namen des Schriftbildes enthält.

Die vorgegebene Schrift muß unter Windows verfügbar sein, entweder über die Initialisierungsdatei oder über einen Aufruf der Funktion *LoadFunc*. Anschließend erscheint ein *DLGITEMTEMPLATE* für jeden Eintrag, der im Dialogfenster dargestellt werden soll. Zeiger auf Dialogfenstervorlagen werden als Parameter den Windows-Funktionen *CreateDialogIndirect*, *CreateDialogIndirectParam*, *DialogBoxIndirect* und *DialogBoxIndirectParam* übergeben.

DLGTEMPLATEHEADER

windows.h

Beschreibung

DLGTEMPLATEHEADER ist keine echte Struktur. Sie kann weder in C++ noch in einer anderen Programmiersprache dargestellt werden, da nullterminierte String-Daten in sie eingebettet sind. Der Header einer Dialogfenstervorlage sollte deshalb als Strukturvorlage, nicht als Struktur betrachtet werden.

Die ersten sechs Einträge dieses Header sind so streng definiert und benannt, daß sie als folgende Struktur erscheinen könnten:

```
struct DialogTemplateHeader
  {
    long dtStyle;
    BYTE dtItemCount;
    int dtX;
    int dtY;
```

```
  int dtCX;
  int dtCY;
};
```

Zusammenfassend läßt sich feststellen, daß der Block mit den Daten der Dialogfenstervorlage an erster Stelle einen long-Wert enthalten muß, der den Stil des Dialogfensters bestimmt (vgl. *DLGTEMPLATE*). Dieser Wert kann eine oder jede Kombination der *DS_*-Dialogfensterstil-Konstanten enthalten. Das folgende Byte *dtItemCount* bestimmt die Anzahl der Einträge im Dialogfenster.

Anschließend bestimmen *dtX* und *dtY* die x- und y-Koordinate der oberen linken Ecke des Dialogfensters. Diese Koordinaten verhalten sich relativ zum Bildschirmursprung, wenn der Stil des Dialogfensters *DS_ABSALIGN* ist. Andernfalls sind diese Koordinaten relativ zum Client-Bereich des übergeordneten Fensters gebildet. *dtCX* und *dtCY* bestimmen Breite und Höhe des Dialogfensters.

Die Einheiten, in denen die Koordinaten und Abmessungen des Dialogfensters festgelegt werden, gehen auf Dialogeinheiten zurück, die aus der Größe der aktuellen Systemschrift abgeleitet werden. Die Funktion *GetDialogBaseUnits* gibt die Größe der Basiseinheit in Pixel zurück. Koordinaten und Abmessungen des Dialogfensters werden dann in Bruchteilen dieser Basiseinheit festgelegt. Horizontale Komponenten (*dtX* und *dtCX*) werden in 1/4 dieser Basiseinheiten, vertikale Komponenten (*dtY* und *dtCY*) in 1/8 angegeben.

Den Koordinaten und der Größe des Dialogfensters folgen drei nullterminierte Strings. *dtMenuName* bestimmt den Namen des Dialogfenstermenüs, *dtClassName* die Fensterklasse des Dialogfensters und *dtCaptionText* dessen Überschrift.

DRAWITEMSTRUCT-Typ

windows.h

Definition

```
typedef struct tagDRAWITEMSTRUCT
  {
    WORD CtlType;
    WORD CtlID;
    WORD itemID;
    WORD itemAction;
    WORD itemState;
    HWND hwndItem;
    HDC hDC;
    RECT rcItem;
    DWORD itemData;
  } DRAWITEMSTRUCT;
```

Beschreibung *DRAWITEMSTRUCT*-Strukturen enthalten Daten zum Zeichnen besitzergezeichneter Steuerelemente. Der Besitzer des

Steuerelements erhält einen Zeiger auf eine *DRAWITEMSTRUCT*-Struktur im Parameter *lParam* der Botschaft *WM_DRAWITEM*.

CtlType ist der Steuerelementtyp, der von einer der *ODT_*-Konstanten festgelegt wird. *CtlID* ist die Steuerelement-ID (nicht für Menüs verwendet) und *itemID* die Menüoptions-ID, abhängig vom Steuerelement. Diese kann für leere Listen oder Kombinationsfenster -1 sein.

itemAction definiert mit den *ODA_*-Konstanten Zeichenaktionen, um festzulegen, wann und wie das Steuerelement gezeichnet werden soll.

itemState beschreibt mit den *ODS_*KONSTANTEN den Status des Eintrags, nachdem er gezeichnet wurde.

hwndItem ist das Fenster-Handle des Steuerelements bzw. für ein Menü das Handle des Menüs, das den Eintrag enthält. *hDC* ist das Handle des Gerätekontextes, der verwendet werden muß, wenn dieses Steuerelement gezeichnet wird.

rcItem ist das umgrenzende Rechteck (*RECT*-Struktur) des Steuerelements im Gerätekontext. Windows schneidet besitzergezeichnete Steuerelemente an dieser Grenze ab, besitzergezeichnete Menüoptionen können diese überschreiten.

itemData kann zwei verschiedene Werte enthalten. Entweder den Wert eines Eintrags in einem besitzergezeichneten Listen- oder Kombinationsfenster, der von einer der Botschaften *CB_ADDSTRING, CB_INSERTSTRING, LB_ADDSTRING* oder *LB_INSERTSTRING* geliefert wurde, oder den *long*-Integer, der im Parameter *NewItem* des Aufrufs von *InsertMenu* an die Menüoption übergeben wurde. *itemData* ist für besitzergezeichnete Schalter nicht definiert.

DWORD-Typ **windows.h**

Definition
```
typedef unsigned long DWORD;
```

Beschreibung Ein vorzeichenloser 32-Bit-Integer.

FARPROC-Typ **windows.h**

Definition
```
typedef int (FAR PASCAL *FARPROC) ();
```

Beschreibung FARPROC ist ein far-Zeiger auf eine Funktion, die Pascal- Aufrufkonventionen verwendet. Es wird ein int zurückgegeben.

GLOBALHANDLE-Typ **windows.h**

Definition
```
typedef HANDLE GLOBALHANDLE;
```

Beschreibung *GLOBALHANDLE* entspricht *HANDLE* und kann verwendet

werden, um im Quelltext kenntlich zu machen, daß es sich um ein Handle auf ein globales Element, wie zum Beispiel einen globalen Speicherblock, handelt.

HANDLE-Typ **windows.h**

Definition
```
typedef WORD HANDLE;
```

Beschreibung *HANDLE* definiert einen allgemeinen Handle-Typ.

HANDLETABLE-Typ **windows.h**

Definition
```
typedef struct tagHANDLETABLE
  {
    HANDLE objectHANDLE[1];
  } HANDLETABLE;
```

Beschreibung *HANDLETABLE* ist ein offenes Array von Handles, das dazu verwendet wird, eine Anzahl von Zeichenwerkzeugen zu speichern. Die Dimensionierung auf ein Array-Element dient hier nur der korrekten Syntax.

HBITMAP-Typ **windows.h**

Definition
```
typedef HANDLE HBITMAP;
```

Beschreibung *HBITMAP* ist ein Handle-Typ für Bitmaps.

HBRUSH-Typ **windows.h**

Definition
```
typedef HANDLE HBRUSH;
```

Beschreibung HBRUSH definiert einen Handle-Typ für Pinselzeichenwerkzeuge.

HCURSOR-Typ **windows.h**

Definition
```
typedef HANDLE HCURSOR;
```

Beschreibung *HCURSOR* definiert einen Handle-Typ für Mauszeiger.

HDC-Typ **windows.h**

Definition
```
typedef HANDLE HDC;
```

Beschreibung *HDC* definiert einen Handle-Typ für Gerätekontexte. Auch Bildschirmkontexte sind Gerätekontexte, so daß deren Handle immer in Variablen des Typs *HDC* gespeichert werden.

HFONT-Typ windows.h

Definition `typedef HANDLE HFONT;`

Beschreibung *HFONT* definiert einen Handle-Typ für Schriftzeichenwerkzeuge.

HICON-Typ windows.h

Definition `typedef HANDLE HICON;`

Beschreibung *HICON* definiert einen Handle-Typ für Symbole.

HMENU-Typ windows.h

Definition `typedef HANDLE HMENU;`

Beschreibung *HMENU* definiert einen Handle-Typ für Menüressourcen.

HPALETTE-Typ windows.h

Definition `typedef HANDLE HPALETTE;`

Beschreibung *HPALETTE* definiert einen Handle-Typ für Paletten.

HPEN-Typ windows.h

Definition `typedef HANDLE HPEN;`

Beschreibung *HPEN* definiert einen Handle-Typ für Stiftzeichenwerkzeuge.

HRGN-Typ windows.h

Definition `typedef HANDLE HRGN;`

Beschreibung *HRGN* definiert einen Handle-Typ für Regionen.

HSTR-Typ windows.h

Definition `typedef HANDLE HSTR;`

Beschreibung *HSTR* definiert einen Handle-Typ für Strings.

HWND-Typ windows.h

Definition `typedef HANDLE HWND;`

Beschreibung *HWND* definiert einen Handle-Typ für Fenster.

Viele API-Funktionsaufrufe erfordern ein Fenster-Handle, um festzulegen, welches Fenster sie bearbeiten sollen.

LOCALHANDLE-Typ **windows.h**

Definition
```
typedef HANDLE LOCALHANDLE;
```

Beschreibung *LOCALHANDLE* entspricht *HANDLE* und kann dazu verwendet werden, im Quelltext kenntlich zu machen, daß es sich um ein Handle zu einem lokalen Speicherblock handelt.

LOGBRUSH-Typ **windows.h**

Definition
```
typedef struct tagLOGBRUSH
{
    WORD lbStyle;
    DWORD lbColor;
    int lbHatch;
  } LOGBRUSH;
```

Beschreibung *LOGBRUSH*-Strukturen werden verwendet, um die Informationen zur Erzeugung eines virtuellen Pinsels mit der Funktion *CreateBrushIndirect* aufzunehmen.

lbStyle enthält eine der *BS_*-Pinselstilkonstanten, die anzeigen, ob der Pinsel kompakt, leer oder schraffiert ist bzw. ein Muster hat.

lbColor ist eine *COLORREF*-Struktur. Die Farbe wird ignoriert, wenn der Stil des Pinsels leer oder gemustert ist. Wenn der Stil *BS_DIBPATTERN* ist, muß das niederwertige WORD eine der *DIB_*-Konstanten enthalten. Diese bestimmen, ob die bezeichneten Farben explizit vorliegen oder einen Index auf die aktuelle Farbpalette angeben.

Das Feld *lbHatch* bestimmt einen Schraffurstil. Abhängig vom Stil des Pinsels kann *lbHatch* folgendes enthalten:

Stil	*lbHatch* enthält
BS_DIBPATTERN	Das Handle eines gepackten geräteunabhängigen Bitmap.
BS_HATCHED	Eine der *HS_*-Schraffurstilkonstanten, die die Ausrichtung der Schraffur anzeigen.
BS_HOLLOW	Ignoriert.
BS_PATTERN	Das Handle des Bitmap, das das Muster definiert.
BS_SOLID	Ignoriert.

LOGFONT-Typ

windows.h

Definition

```
typedef struct tagLOGFONT
  {
    int lfHeight;
    int lfWidth;
    int lfEscapement;
    int lfOrientation;
    int lfWeight;
    BYTE lfItalic;
    BYTE lfUnderline;
    BYTE lfStrikeOut;
    BYTE lfCharSet;
    BYTE lfOutPrecision;
    BYTE lfClipPrecision;
    BYTE lfQuality;
    BYTE lfPitchAndFamily;
    BYTE lfFaceName[LF_FACESIZE];
  } LOGFONT;
```

Beschreibung

LOGFONT-Strukturen enthalten die Attribute einer virtuellen Schrift zur Verwendung durch die Funktion *CreateFontIndirect*.

lfHeight und *lfWidth* bestimmen die durchschnittliche Höhe und Breite der Schrift. *lfEscapement* und *lfOrientation* sind der Schrittschaltungswinkel und der Neigungswinkel des Textes von der x-Achse aus gemessen. Beide werden in Zehntelgraden und entgegen dem Uhrzeigersinn angegeben.

lfWeight bestimmt die Stärke der Schrift in eingeschwärzten Punkten pro Tausend. Der Wert kann zwischen 0 und 1000 liegen. Zum Beispiel gibt 400 einen normalen Grauwert an, 700 bezeichnet eine Fettschrift. Die tatsächlichen Werte variieren in Abhängigkeit vom Schriftbild. Der Wert Null bestimmt eine Standardstärke.

lfItalic, *lfUnderline* und *lfStrikeOut* sind normalerweise 0. Wenn ihr Wert ungleich 0 ist, wird dadurch kursive, unterstrichene oder durchgestrichene Schrift festgelegt. *lfCharSet* ist einer der drei vordefinierten Zeichensätze *ANSI_CHARSET*, *OEM_CHARSET* oder *SYMBOL_CHARSET*. Es können auch andere Zeichensätze definiert werden.

lfOutPrecision enthält eines der *OUT_*-Schriftgenauigkeits-Flags, voreingestellt ist der Wert *OUT_DEFAULT_PRECIS*. *lfClipPrecision* gibt die Clipping-Genauigkeit der Schrift an; voreingestellt ist der Wert *CLIP_DEFAULT_PRECIS*.

lfQuality enthält eines der drei Schriftqualitäts-Flags *DEFAULT_QUALITY*, *DRAFT_QUALITY* oder *PROOF_QUALITY*. *lfPitchAndFamily* ist die Kombination eines Zeichenabstand-Flags (entweder *DEFAULT_PITCH*, *FIXED_PITCH* oder *VARIABLE_PITCH*) und eines Schriftfamilien-Flags (*FF_ROMAN* oder *FF_SCRIPT*). *lfFaceName* enthält die Bezeichnung

der Schrift in einem nullterminierten String. Ist *lfFaceName* NULL, so verwendet die GDI einen Standardschrifttyp.

Die Maximallänge der Schriftbezeichnung (LF_FACESIZE-1) ist 31.

LOGPALETTE-Typ **windows.h**

Definition

```
typedef struct tagLOGPALETTE
  {
    WORD palVersion;
    WORD palNumEntries;
    PALETTEENTRY palPalEntry[1];
  } LOGPALETTE;
```

Beschreibung *LOGPALETTE*-Strukturen enthalten Daten zur Definition einer virtuellen Palette. Sie werden der Funktion *CreatePalette* übergeben.

palVersion bestimmt die Version von Windows für die Struktur, gegenwärtig ist dies 0x0300. *palNumEntries* ist die Anzahl von Einträgen in der Palette, *palPalEntry* ein offenes Array von *PALETTEENTRY*-Strukturen, ein Element für jeden Eintrag in der Palette.

Die Array-Dimensionierung auf ein Element geschieht hier nur der korrekten Syntax wegen.

LOGPEN-Typ **windows.h**

Definition

```
typedef struct tagLOGPEN
  {
    WORD lopnStyle;
    POINT lopnWidth;
    DWORD lopnColor;
  } LOGPEN;
```

Beschreibung *LOGPEN*-Strukturen enthalten die Attribute eines virtuellen Stifts und werden von der Funktion *CreatePenIndirect* verwendet.

Das Feld *lopnStyle* bezeichnet den Stiftstil, wozu eine der *PS_*-Konstanten verwendet wird. *lopnWidth* ist die Breite des Stifts in virtuellen Einheiten, *lopnColor* die Stiftfarbe.

MDICREATESTRUCT-Typ **windows.h**

Definition

```
typedef struct tag MDICREATESTRUCT
  {
    LPSTR szClass;
    LPSTR szTitle;
    HANDLE hOwner;
```

```
    int x, y;
    int cx, cy;
    LONG style;
    LONG lParam;
  } MDICREATESTRUCT;
```

Beschreibung *MDICREATESTRUCT*-Strukturen enthalten Daten für die Erzeugung eines untergeordneten MDI-Fensters. Der Parameter *lParam* einer Botschaft *WM_CREATE* enthält eine *CREATESTRUCT*-Struktur, dessen Feld *lpCreateParams* auf eine *MDICREATESTRUCT*-Struktur zeigt, die von einer Botschaft *WM_-MDICREATE* geliefert wurde.

szClass zeigt auf die Fensterklasse des untergeordneten Fensters, *szTitle* auf den Fenstertitel. *hOwner* ist das Instanz-Handle der Anwendung, die das Fenster erzeugt.

x und *y* sind die anfänglichen x- und y-Koordinaten des untergeordneten Fensters, *cx* und *cy* die anfängliche Breite und Höhe. Sind alle diese Werte auf *CW_USEDEFAULT* gesetzt, so wird ein jeweils passender Standardwert eingesetzt.

style enthält zusätzliche Stile für das Fenster, diese können *WS_MINIMIZE, WS_MAXIMIZE, WS_HSCROLL* oder *WS_-VSCROLL* sein. *lParam* wird von der Anwendung gesetzt.

MEASUREITEMSTRUCT-Typ

windows.h

Definition

```
typedef struct tagMEASUREITEMSTRUCT
  {
    WORD CtlType;
    WORD CtlID;
    WORD itemID;
    WORD itemWidth;
    WORD itemHeight;
    DWORD itemData;
  } MEASUREITEMSTRUCT;
```

Beschreibung *MEASUREITEMSTRUCT*-Strukturen enthalten die Abmessungen eines besitzergezeichneten Steuerelements. Eine *WM_MEASUREITEM*-Botschaft enthält in seinem Parameter *lParam* einen Zeiger auf eine *MEASUREITEMSTRUCT*-Struktur. Wenn deren Elemente nicht richtig belegt werden, arbeiten die besitzergezeichneten Steuerelemente nicht korrekt.

CTLType ist der Steuerelementtyp, der durch eine der *ODT_*-Konstanten bestimmt wird. *CtlID* ist die Steuerelement-ID-Zahl (dies gilt nicht für Menüs). *itemID* ist die Menüoptions-ID für Menüs. Die Felder *itemWidth* und *itemHeight* enthalten die Breite bzw. die Höhe der Option.

Das Feld *itemData* kann zwei Werte enthalten: Entweder einen Eintrag in einem besitzergezeichneten Listen- oder Kombinationsfenster, der von einer der Botschaften *CB_-ADD-*

STRING, *CB_INSERTSTRING*, *LB_ADDSTRING* oder *LB_-INSERTSTRING* geliefert wurde oder den long-Integer, der an die Menüoption über den Parameter *NewItem* des Aufrufs von *InsertMenu* übergeben wurde. *itemData* ist für besitzergezeichnete Schalter undefiniert.

MENUITEMTEMPLATE

windows.h

Definition

```
typedef struct
  {
    WORD mtOption;
    WORD mtID;
    char mtString[1];
  } MENUITEMTEMPLATE
```

Beschreibung Eine MENUITEMTEMPLATE-Struktur enthält Informationen für einen Menüeintrag. Mehrere Strukturen können zu einer Liste von Menüeinträgen verbunden werden. Werden diese dann mit einer MENUITEMTEMPLATEHEADER-Struktur kombiniert, formt diese eine komplette Menüvorlage.

mtOption definiert Typ und Status des Menüeintrags. Es enthält eine oder eine Kombination der *MF_*-Konstanten. *mtID* enthält den ID-Code für den Menüeintrag. Pop-up-Menüeinträge, die ein weiteres Menü anzeigen, haben kein Feld *mtID*. *mtString* ist ein nullterminierter String, der den Namen des Menüeintrags bestimmt.

MENUITEMTEMPLATEHEADER-Typ

windows.h

Definition

```
typedef struct
  {
    WORD versionNumber;
    WORD offset;
  } MENUITEMTEMPLATEHEADER;
```

Beschreibung Die *MENUITEMTEMPLATEHEADER*-Struktur dient als Header für eine Menüvorlage, die auch eine Liste von *MENUITEMTEMPLATE*-Strukturen enthält. *versionNumber* enthält die Versionsnummer der Menüvorlage und sollte Null sein. *offset* bestimmt den Byte-Offset des Anfangs der Eintragsliste, relativ zum Header.

METAFILEPICT-Typ

Definition

```
typedef struct tagMETAFILEPICT
  {
    int mm;
    int xExt;
```

```
  int yExt;
  HANDLE hMF;
} METAFILEPICT;
```

Beschreibung *METAFILEPICT*-Strukturen definieren das Format der MetafileBilder, die verwendet werden, um Metafile-Daten über die Zwischenablage auszutauschen.

mm enthält den Abbildungsmodus, in dem das Bild gezeichnet wird. Wenn der Abbildungsmodus nicht *MM_ISOTROPIC* oder *MM_ANISOTROPIC* ist, bezeichnen *xExt* und *yExt* Breite und Höhe des Rechtecks, in dem das Bild gezeichnet wird. Andernfalls enthalten die Elemente vorgeschlagene Größen (*MM_ANISOTROPIC*) oder relative Größen (*MM_ISOTROPIC*). *hMF* ist ein Handle eines Speicher-Metafiles.

METAHEADER-Typ

windows.h

Definition

```
typedef struct tagMETAHEADER
  {
    WORD mtType;
    WORD mtHeaderSize;
    WORD mtVersion;
    DWORD mtSize;
    WORD mtNoObjects;
    DWORD mtMaxRecord;
    WORD mtNoParameters;
  } METAHEADER;
```

Beschreibung *METAHEADER*-Strukturen definieren das Format des Headers eines Metafiles. Ein Metafile besteht aus einer *METAHEADER*-Struktur, gefolgt von einer Liste von Metafile-Strukturen, die normalerweise vom Typ *METARECORD* sind.

mtType kann zwei Werte enthalten: 1 zeigt an, daß sich das Metafile im Arbeitsspeicher befindet, 2 zeigt an, daß es auf einer Platte abgelegt ist.

mtHeaderSize bestimmt die Größe des Header in WORD. *mtVersion* ist die Windows-Versionsnummer, diese ist derzeit 0x0300. *mtSize* ist die Größe der Datei in Worten.

mtNoObjects kennzeichnet die maximale Anzahl von Objekten, die das Metafile enthalten kann, und *mtMaxRecord* die Größe der längsten Metafile-Struktur in Worten.

mtNoParameters wird gegenwärtig nicht verwendet.

METARECORD-Typ

windows.h

Definition

```
typedef struct tagMETARECORD
{
  DWORD rdSize;
```

```
    WORD rdFunction;
    WORD rdParam[1];
  } METARECORD;
```

Beschreibung Die *METARECORD*-Struktur definiert eine typische Metafile-Struktur. Eine Liste solcher Strukturen folgt auf den Header eines Metafiles.

rdSize bestimmt die Größe der Struktur in Worten.

rdFunction ist die Funktionsnummer, die durch eine *META_*-Konstante bestimmt wird.

rdParm ist ein offenes Array von Parametern des Typs *WORD* für die Funktion, die in umgekehrter Reihenfolge gespeichert sind, in der sie an die Funktion übergeben werden.

Die Array-Dimensionierung auf ein Element ist nur der korrekten Syntax wegen angegeben.

MSG-Typ

windows.h

Definition

```
typedef struct tagMSG
  {
    HWND hwnd;
    WORD message;
    WORD wParam;
    DWORD lParam;
    DWORD time;
    POINT pt;
  } MSG;
```

Beschreibung *MSG*-Strukturen enthalten Botschaftsdaten, die durch Windows an die Anwendungen verteilt werden. Die Informationen werden an die betreffenden Elemente (Fenster) zur Verarbeitung weitergeleitet. Die verschiedenen Elemente enthalten wichtige Informationen für die Anwendung.

Alle Windows-Botschaften mit ihren jeweiligen Parametern sind in Kapitel 2 »Windows-Botschaften« aufgelistet.

hwnd ist das Handle des Fensters, das die Botschaft empfängt. *message* ist die Nummer der Botschaft.

wParam und *lParam* enthalten Informationen für das Fenster. Diese Informationen variieren in Abhängigkeit vom Wert von *message*. Die Werte der Parameter jeder Botschaft sind in Kapitel 2 »Windows-Botschaften« beschrieben.

time enthält die Zeit, zu der die Botschaft übermittelt wurde, *pt* die Bildschirmposition des Mauszeigers zu diesem Zeitpunkt.

MULTIKEYHELP-Typ

windows.h

Definition

```
typedef struct tagMULTIKEYHELP
  {
    WORD mkSize;
    BYTE mkKeyList;
    BYTE szKeyPhrase[1];
  } MULTIKEYHELP;
```

Beschreibung *MULTIKEYHELP*-Strukturen enthalten einen Index auf eine Tabelle von Schlüsselworten für das Windows-Hilfesystem und einen Ausdruck, nach dem gesucht werden soll.

mkSize enthält die Länge der Struktur in Bytes. *mkKeyList* ist ein Zeichen, das bestimmt, welche Schüsselworttabelle durchsucht werden soll, *szKeyPhrase* ein nullterminierter String, der das aufzufindende Schlüsselwort enthält. Er kann beliebig lang sein. Die Dimensionierung auf ein Zeichen ist hier nur der korrekten Syntax wegen angegeben.

OFSTRUCT-Typ

windows.h

Definition

```
typedef struct tagOFSTRUCT
  {
    BYTE cBytes;
    BYTE fFixedDisk;
    WORD nErrCode;
    BYTE reserved[4];
    BYTE szPathName[128];
  } OFSTRUCT;
```

Beschreibung *OFSTRUCT*-Strukturen enthalten Informationen über eine Datei. Sie werden gelesen, wenn die zugehörige Datei geöffnet wird.

cBytes enthält die Länge der Struktur in Bytes. *fFixedDisk* ist ungleich 0, wenn sich die Datei auf einer Festplatte befindet, andernfalls ist es 0. *nErrCode* enthält den DOS-Fehlercode, wenn die Funktion *OpenFile* fehlgeschlagen ist (*OpenFile* liefert dann -1 zurück). *reserved* enthält vier Bytes, die für eine zukünftige Verwendung durch Windows reserviert sind.

szPathName ist ein nullterminierter String, der den vollständigen Pfadnamen der Datei enthält. Er kann nicht länger als 127 Zeichen sein.

PAINTSTRUCT-Typ

windows.h

Definition

```
typedef struct tagPAINTSTRUCT
  {
    HDC hdc;
```

```
    BOOL fErase;
    RECT rcPaint;
    BOOL fRestore;
    BOOL fIncUpdate;
    BYTE rgbReserved[16];
  } PAINTSTRUCT;
```

Beschreibung *PAINTSTRUCT*-Strukturen enthalten Informationen, die von einer Anwendung zum Zeichnen der Client-Bereiche ihrer Fenster verwendet werden. Außer *hdc*, *fErase* und *rcPaint* sind alle anderen Informationen für die interne Verwendung durch Windows reserviert.

hdc ist das Handle des Bildschirmkontexts, in dem die Zeichnung erscheinen soll. *fErase* zeigt an, ob der Hintergrund neu gezeichnet wird, der Wert 0 bedeutet, daß er nicht neu gezeichnet wird. *rcPaint* definiert das Rechteck, in dem die Zeichnung durchgeführt werden soll.

PALETTEENTRY-Typ **windows.h**

Definition

```
typedef struct tagPALETTEENTRY
  {
    BYTE peRed;
    BYTE peGreen;
    BYTE peBlue;
    BYTE peFlags;
  } PALETTEENTRY;
```

Beschreibung Die Struktur *PALETTEENTRY* definiert einen Eintrag in eine virtuelle Palette, so wie von *LOGPALETTE* definiert.

peRed, *peGreen* und *peBlue* repräsentieren jeweils die Intensitäten von Rot, Grün und Blau im Paletteneintrag. 0x00 bezeichnet niedrigste und 0xFF höchste Intensität. *peFlags* enthält Information darüber, wie die Palette eingesetzt werden soll. Es kann 0 oder eines der *PC_*-Flags *PC_EXPLICIT*, *PC_NOCOLLAPSE* oder *PC_RESERVED* sein.

PATTERN-Typ **windows.h**

Definition

```
typedef LOGBRUSH PATTERN;
```

Beschreibung *PATTERN* ist eine andere Bezeichnung für *LOGBRUSH*. Wenn ein virtueller Pinsel für gemustertes Flächenfüllen verwendet wird, kann die Bezeichnung *PATTERN* gewählt werden, um dies zu kennzeichnen.

PINT-Typ

windows.h

Definition

```
typedef int near *PINT;
```

Beschreibung PINT definiert einen Zeiger auf einen vorzeichenbehafteten 16-Bit-Integer.

POINT-Typ

windows.h

Definition

```
typedef struct tagPOINT
  {
    int x;
    int y;
  } POINT;
```

Beschreibung Die Struktur *POINT* bezeichnet die x- und y-Koordinaten (jeweils in *x* und *y*) eines Punktes auf dem Bildschirm oder in einem Fenster.

RECT-Typ

windows.h

Definition

```
typedef struct tagRECT
  {
    int left;
    int top;
    int right;
    int bottom;
  } RECT;
```

Beschreibung Die Struktur *RECT* definiert einen rechteckigen Bereich durch die Vorgabe der oberen linken und der unteren rechten Ecke. *left* und *top* sind die x- und y-Koordinaten der linken oberen Ecke, *right* und *bottom* die x- und y-Koordinaten der unteren rechten Ecke. Das Rechteck darf in Breite und Höhe 32768 Einheiten nicht überschreiten.

RGBQUAD-Typ

windows.h

Definition

```
typedef struct tag RGBQUAD
  {
    BYTE rgbBlue;
    BYTE rgbGreen;
    BYTE rgbRed;
    BYTE rgbReserved;
  } RGBQUAD;
```

Beschreibung *RGBQUAD*-Strukturen enthalten relative RGB-Farbdaten zur Verwendung durch Bitmaps im Feld *bmiColors* einer *BITMAPINFO*-Struktur.

Die Felder *rgbBlue, rgbGreen* und *rgbRed* repräsentieren jeweils die Intensitäten von Blau, Grün und Rot im Bitmap-Pixeleintrag. *rgbReserved* wird nicht verwendet und muß 0 sein.

RGBTRIPLE-Typ

windows.h

Definition

```
typedef struct tag RGBTRIPLE
  {
    BYTE rgbtBlue;
    BYTE rgbtGreen;
    BYTE rgbtRed;
  } RGBTRIPLE;
```

Beschreibung Die Struktur *RGBTRIPLE* bezeichnet die relativen RGB-Farbintensitäten für Bitmaps, so wie das Feld *bmciColors* einer *BITMAPCOREINFO*-Struktur.

Die Felder *rgbtBlue, rgbtGreen* und *rgbtRed* repräsentieren jeweils die Intensitäten von Blau, Grün und Rot im Bitmap-Pixeleintrag.

TEXTMETRIC-Typ

windows.h

Definition

```
typedef struct tagTEXTMETRIC
  {
    int tmHeight;
    int tmAscent;
    int tmDescent;
    int tmInternalLeading;
    int tmExternalLeading;
    int tmAveCharWidth;
    int tmMaxCharWidth;
    int tmWeight;
    int tmItalic;
    BYTE tmUnderlined;
    BYTE tmStruckOut;
    BYTE tmFirstChar;
    BYTE tmLastChar;
    BYTE tmDefaultChar;
    BYTE tmBreakChar;
    BYTE tmPitchAndFamily;
    BYTE tmCharSet;
    int tmOverhang;
    int tmDigitizedAspectX;
    int tmDigitizedAspectY;
  } TEXTMETRIC;
```

Beschreibung Die Struktur *TEXTMETRIC* enthält zahlreiche Felder, die eine physikalische Schrift in Einheiten beschreiben, die vom Abbil-

dungsmodus des Bildschirmkontexts abhängen. Die *TEXTMETRICS*-Strukturen werden von den Funktionen *GetDeviceCaps* und *GetTextMetrics* verwendet.

tmHeight ist die Zeichenhöhe der Schrift, die sich aus der Summe der Oberlänge (*tmAscent*) und der Unterlänge (*tmDescent*) unterhalb der Grundlinie ergibt.

tmInternalLeading und *tmExternalLeading* bestimmen das Ausmaß des zulässigen Leerraums außerhalb des Bereichs von *tmHeight*. *tmInternalLeading* ist der Leerraum innerhalb der Grenzen, *tmExternalLeading* wird zwischen den Textzeilen hinzuaddiert. Beide Felder können auf 0 gesetzt werden.

tmAveCharWidth und *tmMaxCharWidth* bestimmen die durchschnittliche und die maximale Zeichenbreite für die Zeichen in der Schrift.

tmWeight ist die Schriftstärke.

tmItalic, *tmUnderlined* und *tmStruckOut* kennzeichnen jeweils kursiven, unterstrichenen und durchgestrichenen Text.

Der Bereich definierter Zeichen in der Schrift wird von *tmFirstChar* und *tmLastChar* bestimmt, wobei Zeichen außerhalb dieses Bereiches durch das in *tmDefaultChar* festgelegte Zeichen ersetzt werden. *tmBreakChar* ist das Zeichen, das die Wortzwischenräume für Ausrichtungszwecke abgrenzt.

Der Zeichenabstand, die Familie und der Zeichensatz der Schrift werden von *tmPitchAndFamily* und *tmCharSet* definiert. Das niederwertigste Bit in *tmPitchAndFamily* bestimmt den Zeichenabstand der Schrift: fest, wenn das Bit gelöscht, variabel, wenn es gesetzt ist. Die vier höherwertigen Bits bestimmen die Schriftenfamilie, die mit den *FF_*-Schriftenfamilien-Flags gesetzt oder abgefragt werden kann. Der Zeichensatz ist in *tmCharSet* enthalten, hier werden die Schriftzeichensatzflags eingesetzt.

tmOverhang enthält die zusätzliche Breite, die bei manchen synthetisierten Schriften pro String hinzugefügt wird, beispielsweise wenn aus einer normalen eine fette Schrift gemacht wird. Das horizontale und vertikale Seitenverhältnis des Gerätes, für das die Schrift entworfen wurde, ist in *tmDigitizedAspectX* bzw. *tmDigitizedAspectY* abgelegt.

WNDCLASS-Typ

windows.h

Definition

```
typedef struct tagWNDCLASS
  {
    WORD style;
    LONG (FAR PASCAL *lpfnWndProc)
         (HWND, WORD, WORD, LONG);
    int cbClsExtra;
    int cbWndExtra;
    HANDLE hInstance;
```

```
    HICON hIcon;
    HCURSOR hCursor;
    HBRUSH hbrBackground;
    LPSTR lpszMenuName;
    LPSTR lpszClassName;
  } WNDCLASS;
```

Beschreibung *WNDCLASS*-Strukturen enthalten die Attribute einer Fensterklasse – auch bezeichnet als Registrierungsattribute –, so wie sie von der Funktion *RegisterClass* registriert werden.

style enthält den Klassenstil. Möglich ist eine oder eine Kombination der *CS_*-Klassenstilkonstanten.

lpfnWndProc zeigt auf die Fensterfunktion, die Routine, die Botschaften empfängt und verarbeitet.

cbClsExtra ist die Anzahl von Bytes, die am Ende der Struktur *WNDCLASS* angeordnet werden. Diese werden als die Extra-Bytes der Klasse bezeichnet. Auf diese kann mit den Funktionen *GetWindowLong* oder *GetWindowWord* zugegriffen werden, oder sie können mit den Funktionen *SetWindowLong* oder *SetWindowWord* gesetzt werden.

cbWndExtra bestimmt die Anzahl der Bytes, die am Ende der Fensterklasse angeordnet werden sollen.

hInstance ist ein Instanz-Handle, das das Klassenmodul kennzeichnen muß. Es darf nicht 0 sein.

hIcon, hCursor und *hbrBackground* sind jeweils die Handle des Symbols, des Mauszeigers und der Hintergrundfarbe der Klasse. Die Hintergrundfarbe sollte ein Farbwert (eine der Standardsystemfarben, bezeichnet durch eine der um 1 inkrementierten *COLOR_*-Konstanten) oder das Handle eines Pinsels zum Hintergrundzeichnen sein. Wenn *hbrBackground* 0 ist, muß der Hintergrund der Anwendung immer dann gezeichnet werden, wenn ihr Client-Bereich gezeichnet wird. Ob dies notwendig ist, kann durch Bearbeiten der Botschaft *WM_ERASEBKGND* oder durch Abfragen des Elementes *fErase* der von *BeginPaint* erzeugten *PAINTSTRUCT*-Struktur festgestellt werden.

Die Felder *lpszMenuName* und *lpszClassName* zeigen beide auf nullterminierte Strings, die die Ressourcenbezeichnung des Klassenmenüs bzw. die Klassenbezeichnung enthalten.

WORD-Typ

windows.h

Definition `typedef unsigned int WORD;`

Beschreibung Ein vorzeichenloser 16-Bit-Integer. Werte dieses Typs liegen zwischen 0 und 65535.

4. Windows-Stile und -Konstanten

Dieses Kapitel erläutert die häufig verwendeten Konstanten für den Aufruf von Windows-Funktionen und den Empfang von Windows-Botschaften (siehe Kapitel 1 »Windows-Funktionen« und Kapitel 2 »Windows-Botschaften«).

Die Konstanten sind in gebräuchlichen Gruppen alphabetisch angeordnet. Soweit möglich, wurden sie über ihre Präfixe angeordnet und sortiert. So finden sich alle Konstanten, deren Name mit *BI_* beginnt, in der Gruppe »*BI_XXXX*«. Konstanten, die keine Präfixe besitzen, wurden nach ihrer Funktion angeordnet. Die Konstanten *COMPLEXREGION*, *NULLREGION* und *SIMPLEREGION* bestimmen Typen graphischer Regionen und werden als Parameter für dieselben Funktionen verwendet. Sie wurden deshalb in der Gruppe »Region-Flags« angeordnet.

BI_XXXX

Diese Konstanten werden als Werte für das Feld *biCompression* der Struktur *BITMAPINFOHEADER* übergeben, wenn die Funktion *CreateDIBitmap* aufgerufen wird.

BI_RGB	Bitmap wird nicht komprimiert.
BI_RLE8	Bitmap verwendet ein *run-length*-kodiertes Format mit 8 Bit pro Pixel.
BI_RLE4	Bitmap verwendet ein *run-length*-kodiertes Format mit 4 Bit pro Pixel.

BN_XXXX

Die folgenden Benachrichtigungscodes beziehen sich auf Aktionsschalter. Sie werden vom jeweiligen Schalter in einer *WM_COMMAND*-Botschaft übergeben und zeigen eine Aktion des Anwenders an.

BN_CLICKED	Zeigt dem besitzenden Fenster an, daß ein Aktionsschalter betätigt wurde.

BN_DOUBLECLICKED Zeigt dem besitzenden Fenster an, daß der Anwender einen Doppelklick auf einem besitzergezeichneten Aktionsschalter oder einem Schaltfeld durchgeführt hat. Dieser Code bezieht sich nur auf Schalter des Stils *BS_OWNERDRAW* und auf Aktionsschalter.

BS_XXXX

Diese Konstanten bestimmen den Stil des virtuellen Pinsels. Sie werden im Feld *lbStyle* der Struktur *LOGBRUSH* durch die Funktion *CreateBrushIndirect* übergeben.

BS_DIBPATTERN Legt ein Pinselmuster fest, das durch eine geräteunabhängige Bitmap-Beschreibung (DIB) definiert ist.

BS_HATCHED Legt einen Schraffurpinsel fest.

BS_HOLLOW Legt einen Pinsel fest, der mit der Farbe des Hintergrunds malt und dadurch löscht.

BS_NULL Wie *BS_HOLLOW*.

BS_PATTERN Legt einen gemusterten Pinsel fest, der durch ein Speicher-Bitmap definiert ist.

BS_SOLID Legt einen vollständig deckenden Pinsel fest.

Diese Konstanten dienen zur Festlegung der Schalterstile. Sie werden verwendet, wenn Schalter mit *CreateWindow* und *CreateWindowEx* erzeugt werden.

BS_3STATE Dieser Stil erlaubt die Darstellung des Schalters im ausgewählten, nichtausgewählten und im nichtaktivierbaren Zustand.

BS_AUTO3STATE Dieser Schalterstil ist identisch mit *BS_3STATE*. Allerdings ändert der Schalter seinen Zustand, wenn der Anwender ihn anklickt.

BS_AUTOCHECKBOX Identisch mit *BS_CHECKBOX*. Allerdings ändert das Auswahlfeld seinen Zustand, wenn der Anwender es anklickt.

BS_AUTORADIOBUTTON Identisch mit *BS_RADIOBUTTON*, außer, daß das Schaltfeld ausgewählt dargestellt wird, wenn der Anwender es anklickt. Die Anwendung wird mit der Botschaft *BN_CLICKED* verständigt, und die Auswahlmarkierungen werden von allen anderen Schaltfeldern in der Gruppe entfernt.

BS_CHECKBOX Bestimmt ein kleines rechteckiges Auswahlfeld, das angewählt werden kann; seine Umrandung wird fett dargestellt, wenn der Anwender es anklickt. Zugehöriger Text erscheint rechts vom Schalter.

BS_DEFPUSHBUTTON Ein Schalter mit diesem Stil verhält sich ähnlich wie einer mit *BS_PUSHBUTTON*. Allerdings wird der Aktionsschalter mit einer fetten Umrandung dargestellt. Er ist die voreingestellte Antwort des Anwenders, solange keine andere Auswahl mit Maus oder Tastatur getroffen wird. Text wird innerhalb des Schalters angezeigt. Windows sendet eine Botschaft an das übergeordnete Fenster, wenn der Anwender den Schalter anklickt.

BS_GROUPBOX Bestimmt ein Rechteck, in dem andere Schalt- oder Auswahlfelder angeordnet werden. Text wird in der oberen linken Ecke des Rechtecks angezeigt.

BS_LEFTTEXT Bewirkt, daß Text auf der linken Seite des Schaltfeldes oder Auswahlfeldes erscheint. Wird mit den Stilen *BS_CHECKBOX*, *BS_RADIOBUTTON* oder *BS_3STATE* verwendet.

BS_OWNERDRAW Bestimmt einen besitzergezeichneten Schalter. Das übergeordnete Fenster wird benachrichtigt, wenn der Schalter angeklickt wird. Zusätzlich zu den mit der Botschaft *WM_COMMAND* übergebenen normalen Benachrichtigungscodes erhält es Aufforderungen zum Zeichnen, Invertieren oder Deaktivieren des Schalters.

BS_PUSHBUTTON Bestimmt einen Aktionsschalter. Text wird innerhalb des Schalters dargestellt. Das Steuerelement sendet eine Botschaft immer dann an sein übergeordnetes Fenster, wenn der Anwender den Schalter anklickt.

BS_RADIOBUTTON Bestimmt ein kleines kreisförmiges Schaltfeld, das angewählt werden kann; seine Umrandung wird fett dargestellt, wenn der Anwender es anklickt. Text erscheint rechts vom Schaltfeld. Üblicherweise werden zwei oder mehr Schaltfelder zusammengruppiert, um sich gegenseitig ausschließende Auswahlmöglichkeiten darzustellen. Es kann nicht mehr als ein Schaltfeld in der Gruppe zu jeder Zeit ausgewählt sein.

CB_XXXX

Diese Werte werden von Windows-Kombinationsfensterbotschaften (zum Beispiel *CB_ADDSTRING*) übergeben. Ein negativer Wert zeigt einen Fehler an.

CB_ERR	Es ist ein Fehler aufgetreten. Die Aktion konnte nicht durchgeführt werden.
CB_ERRSPACE	Der Platz im Kombinationsfenster reicht zur Durchführung der Aktion nicht aus.
CB_OKAY	Kein Fehler.

CBM_INIT

Die Übergabe von *CBM_INIT* als Parameter *Usage* der Funktion *CreateDIBitmap* zeigt an, daß ein Speicher-Bitmap initialisiert werden soll.

CBN_XXXX

Die folgenden Benachrichtigungscodes werden in den *WM_COMMAND*-Botschaften von Kombinationsfenstern übergeben. Sie zeigen an, daß und welche Aktion stattgefunden hat.

CBN_DBLCLK	Wird an das übergeordnete Fenster übergeben, wenn der Anwender einen String in der Liste des Kombinationsfensters doppelklickt.
CBN_DROPDOWN	Informiert das übergeordnete Fenster eines Kombinationsfensters über die Aktivierung einer Drop-Down-Liste.
CBN_EDITCHANGE	Benachrichtigt das übergeordnete Fenster eines Kombinationsfensters, daß der Anwender den Text im Editierfeld des Kombinationsfensters verändert hat.
CBN_EDITUPDATE	Benachrichtigt das übergeordnete Fenster eines Kombinationsfensters, daß der Text im Editierfeld verändert wurde und neu dargestellt werden muß.
CBN_ERRSPACE	Benachrichtigt das übergeordnete Fenster eines Kombinationsfensters, wenn im System kein Speicher mehr verfügbar ist.
CBN_KILLFOCUS	Benachrichtigt das übergeordnete Fenster eines Kombinationsfensters, daß das Kombinationsfenster den Fokus verloren hat.

CBN_SELCHANGE Benachrichtigt das übergeordnete Fenster eines Kombinationsfensters, daß die Auswahl in der Liste des Kombinationsfensters geändert wurde.

CBN_SETFOCUS Benachrichtigt das übergeordnete Fenster eines Kombinationsfensters, daß das Kombinationsfenster den Fokus erhalten hat.

CBS_XXXX

Diese Konstanten werden zur Festlegung der Kombinationsfensterstile verwendet, wenn Kombinationsfenster mit den Funktionen *CreateWindow* und *CreateWindowEx* erzeugt werden.

CBS_AUTOHSCROLL Dieser Kombinationsfensterstil rollt den Text im Editierfeld automatisch nach links, wenn der Anwender ein Zeichen am Ende der Zeile eintippt. Wenn dieser Stil nicht gesetzt ist, ist nur Text erlaubt, der in die rechteckige Umrandung paßt.

CBS_DROPDOWN Ähnlich wie *CBS_SIMPLE*, außer, daß die Liste nur dann angezeigt wird, wenn der Anwender ein Symbol direkt neben dem Auswahlfeld auswählt.

CBS_DROPDOWNLIST Ähnlich wie *CBS_DROPDOWN*, außer, daß das Editierfeld durch ein statisches Textelement ersetzt wird, das die momentane Auswahl aus der Liste anzeigt.

CBS_HASSTRINGS Dieser Stil kann in Verbindung mit *CBS_OWNERDRAWFIXED* oder *CBS_OWNERDRAWVARIABLE* verwendet werden. Ein besitzergezeichnetes Kombinationsfenster enthält Einträge, die aus Strings bestehen. Das Kombinationsfenster behält den Speicher und die Zeiger für die Strings bei, so daß die Anwendung die Botschaft *LB_GETTEXT* verwenden kann, um den Text für einen bestimmten Eintrag zu liefern.

CBS_NOINTEGRALHEIGHT Dieser Kombinationsfensterstil entspricht der Größe des Kombinationsfensters bei seiner Erzeugung.

Normalerweise wird diese Größe den Einträgen im Kombinationsfenster so angepaßt, daß alle Einträge komplett dargestellt werden können.

CBS_OEMCONVERT Dieser Stil kann in Verwendung mit den Konstanten *CBS_SIMPLE* oder *CBS_DROPDOWN* verwendet werden. Text, der in das Editierfeld eines Kombinationsfensters eingegeben wird, wird vom ANSI-Zeichensatz in den OEM-Zeichensatz und dann zurück in ANSI umgewandelt. Dies sorgt für eine exakte Konvertierung, wenn die Anwendung die Funktion *AnsiToOem* aufruft, um einen ANSI-String im Kombinationsfenster in OEM-Zeichen umzuwandeln.

Dieser Stil ist für Dateinamen enthaltende Kombinationsfenster hilfreich und wird nur für Kombinationsfenster verwendet, die mit den Stilen *CBS_SIMPLE* oder *CBS_DROPDOWN* erzeugt wurden.

CBS_OWNERDRAWFIXED Der Besitzer des Kombinationsfenster ist für die Darstellung der Liste zuständig. Die Einträge in der Liste haben alle die gleiche Höhe.

CBS_OWNERDRAW-VARIABLE Der Besitzer des Kombinationsfensters ist für die Darstellung der Liste zuständig. Die Einträge in der Liste haben eine variable Höhe.

CBS_SIMPLE Die Liste des Kombinationsfensters wird zu jeder Zeit angezeigt. Die aktuelle Auswahl aus der Liste wird im Editierfeld angezeigt.

CBS_SORT Kombinationsfenster dieses Stils besitzen eine sortierte Liste. Strings, die in die Liste eingegeben werden, werden automatisch sortiert. Die Sortierreihenfolge kann für Kombinationsfenster der Stile *CBS_OWNERDRAWFIXED* oder *CBS_OWNERDRAWVARIABLE* voneinander abweichen.

CC_XXXX

Diese Konstanten repräsentieren die Fähigkeiten eines Gerätes, Bögen zu zeichnen.

CC_NONE Gerät kann keine Bögen zeichnen.

CC_CHORD Kreisbögen können dargestellt werden.

CC_CIRCLES Kreise können dargestellt werden.

CC_ELLIPSES Ellipsen können dargestellt werden.

CC_INTERIORS Ausfüllungen können dargestellt werden.

CC_PIE	Kreissegmente können dargestellt werden.
CC_STYLED	Linien verschiedener Stile können dargestellt werden.
CC_WIDE	Breite Linien können dargestellt werden.
CC_WIDESTYLED	Breite Linien verschiedener Stile können dargestellt werden.

CE_XXXX

Diese Flags bestimmen einen Schnittstellenfehler und werden von der Funktion *GetCommError* zurückgegeben.

CE_BREAK	Die Hardware stellt eine Abbruchbedingung fest.
CE_CTSTO	Clear-to-send-Unterbrechung (time out). CTS ist für die durch das Feld *CtsTimeout* der *DCB*-Struktur festgelegte Dauer herabgesetzt, während gleichzeitig versucht wird, ein Zeichen zu übertragen.
CE_DNS	Die parallele Schnittstelle ist nicht ausgewählt.
CE_DSRTO	Data-set-ready-Unterbrechung (time out). DSR ist für die durch das Feld *DsrTimeout* der *DCB*-Struktur festgelegte Dauer herabgesetzt, während gleichzeitig versucht wird, ein Zeichen zu übertragen.
CE_FRAME	Die Hardware stellt einen Framing-Fehler fest.
CE_IOE	Ein I/O-Fehler tritt während des Versuchs der Kommunikation mit einer parallelen Schnittstelle auf.
CE_MODE	Der gewünschte Modus wird nicht unterstützt, oder es wurde ein ungültiger Schnittstellenbezeichner angegeben.
CE_OOP	Der Drucker an einer parallelen Schnittstelle meldet Papierende.
CE_OVERRUN	Empfangenes Zeichen wurde nicht ausgelesen und vom nächsten Zeichen überschrieben. Das Zeichen ist verloren.
CE_PTO	Bei der Kommunikation mit der parallelen Schnittstelle tritt eine Unterbrechung (time out) auf.

CE_RLSDTO	Receive-line-signal-detect-Unterbrechung (time out). RLSD ist für die durch das Feld *RlsTimeout* der *DCB*-Struktur festgelegte Dauer herabgesetzt, während gleichzeitig versucht wird, ein Zeichen zu übertragen.
CE_RXOVER	Überlauf der Empfangswarteschlange. Der Platz in der Empfangswarteschlange reicht nicht aus, oder die Übertragung wurde nach Erhalt des EOF-Zeichens fortgesetzt.
CE_RXPARITY	Die Hardware stellt einen Paritätsfehler fest.
CE_TXFULL	Die Übertragungswarteschlange ist voll. Gleichzeitig wird versucht, ein Zeichen in die Übertragungs-Warteschlange zu schreiben.

CF_XXXX

Diese Konstanten bezeichnen Formate von Daten, die an die Zwischenablage übergeben oder aus ihr entnommen werden können. Außerdem werden die Formate im DDE-Protokoll (dynamischer Datenaustausch) verwendet.

Die Konstanten werden mit den folgenden Funktionen verwendet: *EnumClipboardFormats*, *GetClipboardData*, *GetClipboardFormatName*, *GetPriorityClipboardFormat*, *IsClipboardFormatAvailable* und *SetClipboardData*.

CF_BITMAP	Ein Handle zu einem Windows-Bitmap.
CF_DIB	Ein Speicherblock, der eine geräteunabhängige Bitmap-Beschreibung enthält.
CF_DIF	Software Arts' Data Interchange Format.
CF_DSPBITMAP	Anzeigeinformation eines privaten Bitmap-Formats.
CF_DSPMETAFILEPICT	Anzeigeinformation eines privaten Metafile-Formats.
CF_DSPTEXT	Anzeigeinformation eines privaten Textformats.
CF_METAFILEPICT	Metafile-Bildformat in einer *METAFILEPICT*-Datenstruktur
CF_OEMTEXT	Textformat, das Zeichen aus dem OEM-Zeichensatz enthält. Jede Zeile endet mit einer Kombination Wagenrücklauf/Zeilenschaltung (CR-LF). Ein Null-Zeichen kennzeichnet das Datenende.

CF_OWNERDISPLAY Besitzeranzeigeformat. Der Besitzer der Zwischenablage muß das Anwendungsfenster der Zwischenablage anzeigen und aktualisieren und empfängt die Botschaften *WM_ASKCBFORMATNAME*, *WM_HSCROLLCLIPBOARD*, *WM_PAINTCLIPBOARD*, *WM_SIZECLIPBOARD* und *WM_VSCROLLCLIPBOARD*.

CF_PALETTE Handle zu einer Farbpalette. Immer, wenn eine Anwendung Daten in der Zwischenablage speichert, die von einer Farbpalette abhängen oder eine solche voraussetzen, dann sollte sie die Palette ebenfalls in der Zwischenablage speichern.

Wenn die Zwischenablage Daten im Format *CF_PALETTE* (virtuelle Farbpalette) enthält, sollte die Anwendung davon ausgehen, daß andere Daten in der Zwischenablage unabhängig von dieser virtuellen Palette realisiert sind. Die Zwischenablage-Viewer-Anwendung (CLIPBRD.EXE) verwendet als aktuelle Palette ein Objekt im Format *CF_PALETTE* in der Zwischenablage, wenn die anderen Formate in der Zwischenablage dargestellt werden.

CF_PRIVATEFIRST bis *CF_PRIVATELAST* Bereich von Integerwerten, die für private Formate benutzt werden. Daten-Handle, die Formaten in diesem Bereich zugeordnet sind, werden nicht automatisch freigegeben.

Alle Daten-Handle müssen von der Anwendung freigegeben werden, bevor die Anwendung beendet oder wenn eine Botschaft *WM_DESTROYCLIPBOARD* empfangen wird.

CF_SYLK Microsoft-Symbolic-Link (SYLK) Format.

CF_TEXT Textformat. Jede Zeile endet mit einer Kombination Wagenrücklauf/Zeilenschaltung (CR-LF). Ein Null-Zeichen kennzeichnet das Datenende.

CF_TIFF Tag-Image-File-Format.

CCHDEVICENAME

Die Konstante *CCHDEVICENAME* begrenzt die Länge eines Gerätenamensstrings auf 31 Zeichen (ohne abschließendes Nullzeichen).

CLIP_XXXX

Diese Konstanten bestimmen die Clipping-Präzision von Schriften, die mit der Funktion *CreateFont* erzeugt werden. Sie steuern die Textdarstellung, wenn der Text teilweise außerhalb der Clipping-Region liegt.

CLIP_CHARACTER_PRECIS
CLIP_DEFAULT_PRECIS
CLIP_STROKE_PRECIS

COLOR_XXXX

Diese Codes legen die Eigenschaften der Windows-Anwenderschnittstelle fest. Programmierer können deren Farben direkt mit der Funktion *SetSysColor* setzen.

COLOR_ACTIVEBORDER	Umrahmung des aktiven Fensters.
COLOR_ACTIVECAPTION	Überschrift des aktiven Fensters.
COLOR_APPWORKSPACE	Hintergrundfarbe von Multidokumentschnittstellenrahmenfenstern (MDI).
COLOR_BACKGROUND	Windows-Desktop.
COLOR_BTNFACE	Flächenschattierung von Aktionsschaltern.
COLOR_BTNSHADOW	Rahmenschattierung von Aktionsschaltern.
COLOR_BTNTEXT	Text in Aktionsschaltern.
COLOR_CAPTIONTEXT	Text in Überschrift, Bildgrößenfeld und Bildlaufpfeilfeld.
COLOR_GRAYTEXT	Grauer (nichtaktivierbarer) Text. Diese Farbe wird auf 0 gesetzt, wenn der momentane Bildschirmtreiber einen echten Grauton nicht unterstützt.
COLOR_HIGHLIGHT	Gewählte Option in einem Steuerelement.
COLOR_HIGHLIGHTTEXT	Text einer gewählten Option in einem Steuerelement.
COLOR_INACTIVEBORDER	Umrahmung eines deaktivierten Fensters.
COLOR_INACTIVECAPITON	Überschrift eines deaktivierten Fensters.
COLOR_MENU	Hintergrund eines Zeilenmenüs.
COLOR_MENUTEXT	Text in Zeilenmenüs.
COLOR_SCROLLBAR	Hintergrund einer Bildlaufleiste.
COLOR_WINDOW	Fensterhintergrund.

COLOR_WINDOWFRAME Fensterrahmen.

COLOR_WINDOWTEXT Text in Fenstern.

CP_XXXX

Diese Konstanten zeigen die Clipping-Fähigkeiten eines Geräts an.

CP_NONE Kein Clipping der Ausgabe möglich.

CP_RECTANGLE Clipping der Ausgabe in Rechtecke möglich.

CS_XXXX

Diese Konstanten der Fensterklassenstile werden im Feld *style* der Datenstruktur *WNDCLASS* verwendet. Sie können mit dem ODER-Operator (|) verknüpft werden.

CS_BYTEALIGNCLIENT Ausrichtung des Client-Bereichs eines Fensters auf eine Byte-Grenze in der x-Richtung.

CS_BYTEALIGNWINDOW Ausrichtung eines Fenster auf eine Byte-Grenze in der x-Richtung.

CS_CLASSDC Gibt der Fensterklasse einen eigenen Gerätekontext, den sich ihre Instanzen teilen.

CS_DBLCLKS Fenster erhält Doppelklickbotschaften.

CS_GLOBALCLASS Bestimmt, daß die Fensterklasse eine globale Klasse der Anwendung ist. Die globale Klasse einer Anwendung wird von einer Anwendung oder Bibliothek erzeugt und ist für alle Anwendungen verfügbar. Die Klasse wird zerstört, wenn die Anwendung oder Bibliothek, die die Klasse erzeugte, beendet wird. Es ist deshalb notwendig, daß alle Fenster, die mit der anwendungsglobalen Klasse erzeugt wurden, vor diesem Ereignis geschlossen werden.

CS_HREDRAW Neuzeichnung des kompletten Fensters, wenn sich dessen Breite ändert.

CS_NOCLOSE Sperrt die Schließoption im Systemmenü des Fensters.

CS_OWNDC Gibt jeder Fensterinstanz ihren eigenen Gerätekontext. Der *CS_OWNDC*-Stil ist zwar grundsätzlich uneingeschränkt verwendbar, aller-

	dings benötigt jeder Gerätekontext ungefähr 800 Byte Speicher.
CS_PARENTDC	Das Fenster verwendet den Gerätekontext seines übergeordneten Fensters.
CS_SAVEBITS	Sichert einen Bildschirmbereich, der von einem Fenster verdeckt wird. Windows verwendet das gesicherte Bitmap, um den Bildschirmbereich wiederherzustellen, wenn ein Fenster verschoben wird. Windows stellt das Bitmap an seiner ursprünglichen Position dar und sendet keine *WM_PAINT*-Botschaft an Fenster, die durch das Fenster verdeckt wurden, wenn der durch das Bitmap verwendete Speicher nicht freigegeben wurde und wenn andere Bildschirmaktionen das gespeicherte Bild nicht ungültig gemacht haben. Eine Anwendung sollte dieses Bit nur für kleine Fenster setzen, die nur kurz angezeigt und dann beseitigt werden, bevor andere Bildschirmaktivitäten stattfinden. Das Setzen dieses Bits für ein Fenster erhöht die Zeitdauer für die erste Darstellung des Fensters wegen der benötigten Zeit für die Speicherreservierung und Speicherung des Bitmap.
CS_VREDRAW	Neuzeichnung des kompletten Fensters, wenn sich dessen Höhe ändert.

CTLCOLOR_XXXX

Diese Flags zeigen den Typ des Steuerelements oder Dialogfensters an, das dargestellt werden soll. Sie werden mit der Botschaft *WM_CTLCOLOR* übergeben.

CTLCOLOR_BTN	Schalter.
CTLCOLOR_DLG	Dialogfenster.
CTLCOLOR_EDIT	Editierfeld.
CTLCOLOR_LISTBOX	Liste.
CLTCOLOR_MAX	Vergrößerungsfeld.
CTLCOLOR_MSGBOX	Meldungsfenster.
CTLCOLOR_SCROLLBAR	Bildlaufleiste.
CTLCOLOR_STATIC	Statisches Steuerelement.

CW_USEDEFAULT

CW_USEDEFAULT wird als Parameter in den Funktionen *CreateWindow* oder *CreateWindowEx* verwendet. Sie veranlaßt Windows, dem neuerstellten Fenster eine Standardgröße bzw. -position zuzuweisen.

DCB_XXXX

Diese *Flags* können als Bit-Maske verwendet werden, um die Bitfeldeinstellungen im Feld *Flags* einer *DCB*-Struktur zu liefern oder zu setzen.

Konstante	Geliefertes Bit
DCB_BINARY	*fBinary*
DCB_CHEVT	*fChEvt*
DCB_DTRDISABLE	*fDtrDisable*
DCB_DTRFLOW	*fDtrFlow*
DCB_INX	*fInX*
DCB_NULL	*fNull*
DCB_OUTX	*fOutX*
DCB_OUTXCTSFLOW	*fOutxCtsFlow*
DCB_OUTXDSRFLOW	*fOutxDsrFlow*
DCB_PARITY	*fParity*
DCB_PECHAR	*fPeChar*
DCB_RTSDISABLE	*fRtsDisable*
DCB_RTSFLOW	*fRtsflow*

DIB_XXXX

Diese Konstanten bestimmen den Zugriff auf Farben für die Verwendung in geräteunabhängigen Bitmaps. Sie werden in den Funktionen *CreateDIBitmap*, *CreateDIBPatternBrush*, *GetDIBits*, *SetDIBits*, *SetDIBitsToDevice* und *StretchDIBits* verwendet.

DIB_PAL_COLORS	Die Farbtabelle ist ein Array von 16-Bit-Indexen in die aktuell realisierte virtuelle Palette.
DIB_RGB_COLORS	Die Farbtabelle enthält explizite RGB-Farbwerte.

DLGC_XXXX

Diese Codes legen fest, welche Dialogeingaben eine Anwendung anstelle von Windows bearbeitet. Sie werden von *WM_GETDLGCODE* zurückgegeben.

DLGC_BUTTON	Aktionsschalterbotschaften.
DLGC_DEFPUSHBUTTON	Botschaften des Standardaktionsschalters.
DLGC_HASSETSEL	*EM_SETSEL*-Botschaften.
DLGC_RADIOBUTTON	Schaltfeldbotschaften.
DLGC_UNDEFPUSHBUTTON	Aktionsschalterbotschaften.
DLGC_WANTALLKEYS	Alle Tastatureingabebotschaften.
DLGC_WANTARROWS	Cursor-Tastenbotschaften.
DLGC_WANTCHARS	*WM_CHAR*-Botschaften.
DLGC_WANTMESSAGE	Alle Tastatureingaben (die Anwendung leitet die Botschaft an das Steuerelement weiter).
DLGC_WANTTAB	Tab-Tastenbotschaften.

DLGWINDOWEXTRA

Soll eine neue Klasse für Dialogfenster aus einer Ressource definiert werden, wird das Feld *cbWndExtra* der Struktur *WNDCLASS* auf *DLGWINDOWEXTRA* gesetzt.

DM_XXXX

Diese Konstanten werden als Feldauswahlbits verwendet. Jede Konstante repräsentiert ein bestimmtes Bit im Feld *dmFields* einer *DEVMODE*-Struktur. Ist ein bestimmtes Bit gesetzt, zeigt es an, daß das korrespondierende Feld in der Struktur initialisiert wurde.

Konstante	Feld
DM_COLOR	*dmColor*
DM_COPIES	*dmCopies*
DM_DEFAULTSOURCE	*dmDefaultSource*
DM_DUPLEX	*dmDuplex*
DM_ORIENTATION	*dmOrientation*
DM_PAPERLENGTH	*dmPaperLength*
DM_PAPERSIZE	*dmPaperSize*
DM_PAPERWIDTH	*dmPaperWidth*
DM_PRINTQUALITY	*dmPrintQuality*
DM_SCALE	*dmScale*

Diese Konstanten werden im Parameter *Mode* der Funktion *ExtDeviceMode* verwendet, um bestimmte Operationen zu veranlassen.

Mode kann einen oder mehrere der folgenden Werte annehmen.

DM_COPY	Schreibt die Gerätetreibereinstellungen in die *DEVMODE*-Struktur, die im Parameter *DevModeOutput* übergeben wird.
DM_MODIFY	Verändert die Einstellungen des Gerätetreibers und verwendet Informationen der *DEVMODE*-Struktur, die in *DevModeInput* übergeben wird.
DM_PROMPT	Erzeugt das Dialogfenster für die Gerätetreibereinstellung und stellt den Drucker nach Maßgabe der Eingaben des Anwenders ein.
DM_UPDATE	Aktualisiert die Druckerumgebung sowie die Datei WIN.INI gemäß der aktuellen Druckereinstellungen.

DMBIN_XXXX

Diese Konstanten werden im Feld *dmDefaultSource* einer *DEVMODE*-Struktur verwendet. Sie legen fest, welcher Papierschacht angesprochen wird.

DMBIN_AUTO	*DMBIN_LARGEFMT*	*DMBIN_SMALLFMT*
DMBIN_CASSETTE	*DMBIN_LOWER*	*DMBIN_TRACTOR*
DMBIN_ENVELOPE	*DMBIN_MANUAL*	*DMBIN_UPPER*
DMBIN_ENVMANUAL	*DMBIN_MIDDLE*	
DMBIN_LARGECAPACITY	*DMBIN_ONLYONE*	

DMCOLOR_XXXX

Diese Konstanten werden im Feld *dmColor* einer *DEVMODE*-Struktur verwendet. Sie zeigen an, ob der Drucker farbig drucken kann.

DMCOLOR_MONOCHROME	Drucker kann nur monochrom drucken
DMCOLOR_COLOR	Drucker druckt farbig.

DMDUP_XXXX

Diese Konstanten werden im Feld *dmDuplex* einer *DEVMODE*-Struktur verwendet. Sie legen die Art des Duplex-Druckes (doppelseitig) fest.

DMDUP_HORIZONTAL	
DMDUP_SIMPLEX	Nur einseitig drucken.
DMDUP_VERTICAL	

DMORIENT_XXXX

Diese Konstanten werden im Feld *dmOrientation* einer *DEVMODE*-Struktur verwendet. Sie bestimmen die gewünschte Druckorientierung.

DMORIENT_PORTRAIT	Portrait-Orientierung (Hochformat).
DMORIENT_LANDSCAPE	Landscape-Orientierung (Querformat).

DMPAPER_XXXX

Diese Konstanten werden im Feld *dmPaperSize* einer *DEVMODE*-Struktur verwendet. Sie bestimmen die Größe des verwendeten Papiers.

Konstante	Breite	Höhe	Einheit
DMPAPER_10X14	10	14	Zoll
DMPAPER_11X17	11	17	Zoll
DMPAPER_A3	297	420	mm
DMPAPER_A4	210	297	mm
DMPAPER_A4SMALL	210	297	mm
DMPAPER_A5	148	210	mm
DMPAPER_B4	250	354	mm
DMPAPER_B5	182	257	mm
DMPAPER_CSHEET	C		
DMPAPER_DSHEET	D		
DMPAPER_ENV_10	4 1/8	9 1/2	Zoll
DMPAPER_ENV_11	4 1/2	10 3/8	Zoll
DMPAPER_ENV_12	4 3/4	11	Zoll
DMPAPER_ENV_14	5	11 1/2	Zoll
DMPAPER_ENV_9	3 7/8	8 7/8	Zoll
DMPAPER_ESHEET	E		
DMPAPER_EXECUTIVE	7 1/2	10	Zoll
DMPAPER_FOLIO	8 1/2	13	Zoll
DMPAPER_LEDGER	17	11	Zoll
DMPAPER_LEGAL	8 1/2	14	Zoll
DMPAPER_LETTER	8 1/2	11	Zoll
DMPAPER_LETTERSMALL	8 1/2	11	Zoll
DMPAPER_NOTE	8 1/2	11	Zoll
DMPAPER_QUARTO	215	275	mm
DMPAPER_STATEMENT	5 1/2	8 1/2	Zoll
DMPAPER_TABLOID	11	17	Zoll

DMRES_XXXX

Diese Konstanten werden im Feld *dmPrintQuality* einer *DEVMODE*-Struktur verwendet. Sie bestimmen die beim Druck verwendete Auflösung.

DMRES_DRAFT	Entwurfsmodus.
DMRES_HIGH	Hohe Druckqualität.
DMRES_LOW	Niedrige Druckqualität.
DMRES_MEDIUM	Mittlere Druckqualität.

DRIVE_XXXX

Diese Konstanten bestimmen den Typ eines Laufwerks und werden von der Funktion *GetDriveType* zurückgegeben.

DRIVE_FIXED	Festplattenlaufwerk.
DRIVE_REMOTE	Netzlaufwerk.
DRIVE_REMOVEABLE	Diskettenlaufwerk.

Drucker-Escape-Codes

Diese Codes werden in der *Escape*-Funktion übergeben, um Anwendungen den direkten Zugriff auch auf diejenigen gerätespezifischen Fähigkeiten zu ermöglichen, die von der grafischen Geräteschnittstelle nicht unterstützt werden.

ABORTDOC	Bricht den aktuellen Druckauftrag ab.
BANDINFO	Kopiert Informationen über die speziellen Fähigkeiten eines Geräts in eine Struktur, die an die Funktion *Escape* übergeben wird. Diese Fähigkeiten beziehen sich auf die Möglichkeit, die Ausgabe in einzelne Abschnitte aufzuteilen, die vor der Ausgabe in einem Metafile gespeichert wird.
BEGIN_PATH	Öffnet einen Pfad, der ein Polygon oder einen Linienzug aus verbundenen Punkten darstellt. Pfade werden verwendet, um die Befehlsübergabe an PostScript-Drucker zu erleichtern.
CLIP_TO_PATH	Definiert eine Clipping-Region auf Grundlage des aktuellen Pfads.

DEVICEDATA	Wie *PASSTHROUGH*.
DRAFTMODE	Schaltet den Entwurfsmodus (EDV-Qualität) an oder aus.
DRAWPATTERNRECT	Zeichnet ein gemustertes, ein schwarzes oder ein Graustufenrechteck auf einem PCL-kompatiblen Hewlett-Packard Drucker.
ENABLEDUPLEX	Schaltet die Druckerfunktion zum beidseitigen Druck ein.
ENABLEPAIRKERNING	Aktiviert oder deaktiviert die Kerning-Funktion des Druckers.
ENABLERELATIVEWIDTHS	Aktiviert oder deaktiviert die relative Zeichenbreite. Ist diese Option deaktiviert, ist die Breite eines Strings identisch mit der Summe der Zeichenbreiten. Ist die Option aktiviert, muß auf die Breitentabelle der Schrift zugegriffen werden, um die Breite des Strings zu berechnen.
END_PATH	Beendet einen Pfad (vgl. *BEGIN_PATH*).
ENDDOC	Beendet einen Druckauftrag, der durch *STARTDOC* gestartet wurde (vgl. *STARTDOC*).
ENUMPAPERBINS	Liefert Informationen über die Papierschächte des Druckers. Es sollte besser die Konstante *GETSETPAPERBINS* verwendet werden. Dieser *Escape*-Code wurde nur aus Gründen der Abwärtskompatibilität übernommen.
ENUMPAPERMETRICS	Liefert die Anzahl unterstützter Papiertypen oder die Größe der druckbaren Rechtecke. Es sollte besser die Funktion *ExtDeviceMode* verwendet werden. Dieser *Escape*-Code wurde nur aus Gründen der Abwärtskompatibilität übernommen.
EPSPRINTING	Unterdrückt den Steuerungsabschnitt des PostScript-Headers und unterbindet dadurch sämtliche GDI-Aufrufe.
EXT_DEVICE_CAPS	Liefert Informationen über gerätespezifische Druckerfähigkeiten, die über die Rückgabe der Funktion *GetDeviceCaps* hinausgehen.
EXTTEXTOUT	Entspricht der Funktion *ExtTextOut*.
FLUSHOUTPUT	Leert den Gerätepuffer.
GETCOLORTABLE	Liefert einen RGB-Farbtabelleneintrag.
GETEXTENDED-TEXTMETRICS	Liefert die erweiterten Textabmessungen der ausgewählten Schrift.

GETEXTENTTABLE	Liefert die Ausdehnung (Breite) eines Zeichenbereichs im aktuellen Zeichensatz der Schrift.
GETPAIRKERNTABLE	Liefert die Zeichenpaar-Kerning-Tabelle.
GETPHYSPAGESIZE	Liefert die physikalische Seitengröße.
GETPRINTINGOFFSET	Liefert den Offset der oberen linken Position auf einer Seite, an der das Drucken oder Zeichnen begonnen werden soll.
GETSCALINGFACTOR	Liefert den Skalierungsfaktor des Druckers in x- und y-Richtung.
GETSETPAPERBINS	Liefert die Anzahl der Papierschächte des Druckers und setzt den aktuellen Schacht ein.
GETSETPAPERMETRICS	Liefert die Papierabmessungsinformationen des Druckers und stellt sie neu ein. Es sollte besser die Funktion *ExtDeviceMode* verwendet werden. Dieser *Escape*-Code wurde nur aus Gründen der Abwärtskompatibilität übernommen.
GETSETPRINTORIENT	Liefert die aktuelle Papierausrichtung oder stellt sie neu ein. Es sollte besser die Funktion *ExtDeviceMode* verwendet werden. Dieser *Escape*-Code wurde nur aus Gründen der Abwärtskompatibilität übernommen.
GETTECHNOLOGY	Liefert einen String, der die grundlegende Arbeitsweise des Druckers bezeichnet, z.B. PostScript.
GETTRACKKERNTABLE	Liefert die Track-Kerning-Tabelle der aktuell ausgewählten Schrift.
GETVECTORBRUSHSIZE	Liefert die Breite des zum Ausfüllen geschlossener Formen verwendeten Plotterstifts in Geräteeinheiten.
GETVECTORPENSIZE	Liefert die Breite des für Schraffurmuster verwendeten Plotterstifts in Geräteeinheiten.
MFCOMMENT	Fügt einen Kommentar zu einem Metafile hinzu.
NEWFRAME	Weist den Drucker an, auf einer neuen Seite fortzufahren.
NEXTBAND	Teilt dem Gerätetreiber mit, daß die Ausgabe eines Abschnitts in ein Metafile abgeschlossen wurde.
PASSTHROUGH	Erlaubt einer Anwendung die unmittelbare Übergabe von Daten an den Drucker.

QUERYESCSUPPORT	Ermittelt, ob der Gerätetreiber einen bestimmten *Escape*-Code unterstützt.
RESTORE_CTM	Stellt die vorher gesicherte, aktuelle Transformationsmatrix wieder her.
SAVE_CTM	Sichert die aktuelle Transformationsmatrix.
SELECTPAPERSOURCE	Wird von *GETSETPAPERBINS* ersetzt und wurde nur aus Gründen der Abwärtskompatibilität übernommen.
SET_ARC_DIRECTION	Bestimmt die Richtung, in der Kreisbögen durch die Funktion *Arc* gezeichnet werden.
SET_BACKGROUND_COLOR	Liefert die Hintergrundfarbe eines Geräts oder stellt sie neu ein.
SET_BOUNDS	Stellt das eingrenzende Rechteck für das darzustellende Bild ein.
SET_CLIP_BOX	Setzt ein Clipping-Rechteck oder stellt das vorhergehende wieder her.
SET_POLY_MODE	Stellt den Poly-Modus ein, der festlegt, wie Aufrufe der Funktionen *Polygon* und *Polyline* interpretiert werden.
SET_SCREEN_ANGLE	Setzt den aktuellen Neigungswinkel der Rasterung und simuliert so die Neigung eines photographischen Farbfilters, um die Farbtrennung für eine Grundfarbe zu ermöglichen.
SET_SPREAD	Setzt den Umfang, um den nichtweiße Grundelemente erweitert werden, damit Fehler bei der Reproduktion ausgeglichen werden.
SETABORTPROC	Setzt die Abbruchfunktion für den aktuellen Druckauftrag.
SETALLJUSTVALUES	Setzt Ausrichtungswerte für die Textausgabe.
SETCOLORTABLE	Setzt einen RGB-Farbtabelleneintrag.
SETCOPYCOUNT	Bestimmt die Anzahl der Kopien, die von jeder Seite gedruckt werden sollen.
SETKERNTRACK	Bestimmt, welcher Kerning-Track verwendet werden soll.
SETLINEJOIN	Legt fest, wie sich schneidende Linien miteinander verbunden werden: mit abgerundeten, eckigen oder bis zur Spitze verlängerten Ecken.
SETMITERLIMIT	Setzt den Gehrungswinkel, ab dem Linienverbindungen nicht bis zur Spitze verlängert, sondern eckig abgeschnitten werden.

STARTDOC	Teilt dem Gerätetreiber mit, daß ein neuer Druckauftrag beginnt.
STRETCHBLT	Entspricht der Funktion *StretchBlt*.
TRANSFORM_CTM	Verändert die aktuelle Transformationsmatrix.

DS_XXXX

Diese Konstanten werden verwendet, um den Stil eines Dialogfensters festzulegen, wenn es mit *CreateWindow* oder *CreateWindowEx* erzeugt wird.

DS_ABSALIGN	Bestimmt, daß das Dialogfenster relativ zum Bildschirmursprung und nicht zum Besitzer des Dialogfensters ausgerichtet wird.
DS_LOCALEDIT	Bestimmt, daß Textspeicher für Editierfelder im lokalen Datensegment einer Anwendung reserviert wird. Dies erlaubt die Nutzung der Botschaften *EM_GETHANDLE* und *EM_SETHANDLE*. Wenn dieser Stil nicht vorgegeben ist, befindet sich der Datenbereich für Editierfelder in einem separaten globalen Datenblock.
DS_MODALFRAME	Bestimmt ein Dialogfenster mit modalem Rahmen. Dieser Stil kann mit *WS_CAPTION* und *WS_SYSMENU* kombiniert werden, um ein Dialogfenster mit einer Überschrift und einem Systemmenü zu erzeugen.
DS_NOIDLEMSG	Bestimmt, daß Windows keine *WM_ENTERIDLE*-Botschaft an den Besitzer übergibt, während das Dialogfenster angezeigt wird.
DS_SETFONT	Bestimmt, daß eine andere Schrift als die Systemschrift verwendet wird. Ist dieses Flag gesetzt, muß das Dialogfenster auf eine erweiterte *DLGTEMPLATE*-Datenstruktur zugreifen. Diese Struktur besteht aus dem Header einer *DLGTEMPLATE*-Struktur, gefolgt von einer Struktur mit Schriftinformationen. Wenn Windows ein Dialogfenster mit diesem Attribut erzeugt, sendet Windows die Botschaft *WM_SETFONT* vor der Erstellung der Steuerelemente an das Dialogfenster.
DS_SYSMODAL	Bestimmt ein systemmodales Fenster. Alle Windows-Anwendungen werden angehalten, während dieses Dialogfenster angezeigt wird.

DT_XXXX

Folgende Konstanten repräsentieren den Typ des verwendeten Geräts.

DT_CHARSTREAM	Zeichen-Stream.
DT_DISPFILE	Bildschirmdatei.
DT_METAFILE	Metafile.
DT_PLOTTER	Vektorplotter.
DT_RASCAMERA	Rasterkamera.
DT_RASDISPLAY	Rasterbildschirm.
DT_RASPRINTER	Rasterdrucker.

Folgende Konstanten legen die Formatierungsoptionen für die Funktion *DrawText* fest.

DT_BOTTOM	Bestimmt an der Grundlinie ausgerichteten Text. Dieser Wert muß mit *DT_SINGLELINE* kombiniert werden.
DT_CALCRECT	Ermittelt Höhe und Breite des Rechtecks. Gibt es mehrere Textzeilen, verwendet *DrawText* die Breite des durch den Parameter *Rect* bezeichneten Rechtecks und erweitert dessen Basis, um auch die letzte Textzeile einzuschließen. Existiert nur eine Textzeile, wird die rechte Seite des Rechtecks so angepaßt, daß auch das letzte Zeichen der Textzeile eingeschlossen wird. In jedem Fall gibt *DrawText* die Höhe des Textes zurück, stellt ihn aber nicht dar.
DT_CENTER	Zentriert den Text horizontal.
DT_EXPANDTABS	Erweitert die Tabulatoren. Der Vorgabewert von Zeichen je Tabulator ist acht.
DT_EXTERNALLEADING	Schließt die Oberlinien der Schrift ein. Normalerweise werden diese Oberlinien in die Höhe einer Textzeile nicht eingeschlossen.
DT_LEFT	Stellt Text linksbündig dar.
DT_NOCLIP	Zeichnet ohne Clipping. DrawText arbeitet schneller, wenn *DT_NOCLIP* verwendet wird.
DT_NOPREFIX	Schaltet die Bearbeitung von Präfix-Zeichen aus. Normalerweise interpretiert *DrawText* das mnemonische Präfix-Zeichen »&« als Anweisung zum Unterstreichen des nachfolgenden Zeichens. Das mnemonische Präfix-Zeichen »&&« wird dann als Anweisung zum Druck ei-

nes einzelnen unterstrichenen »&« interpretiert.

Das Setzen von *DT_NOPREFIX* schaltet diese Bearbeitung aus.

DT_RIGHT Richtet Text rechtsbündig aus.

DT_SINGLELINE Legt eine einzige Textzeile fest. Wagenrücklauf und Zeilenvorschub führen nicht zum Zeilenumbruch.

DT_TABSTOP Setzt Tabulatoren. Das höherwertige Byte des Parameters *Format* bestimmt die Anzahl der Zeichen pro Tabulator. Der Standardvorgabewert ist acht.

DT_TOP Bestimmt bei einzelnen Textzeilen die Ausrichtung des Textes an der Oberlinie.

DT_VCENTER Legt bei einzelnen Textzeilen vertikale Zentrierung fest.

DT_WORDBREAK Legt den Zeilenumbruch fest: Zeilen werden wortweise umbrochen, wenn ein Wort nicht mehr in die vorhergehende Zeile paßt. Dabei wird die durch den Parameter *Rect* festgelegte Größe des Rechtecks zugrunde gelegt.

Auch eine Wagenrücklauf-Zeilenvorschub-Sequenz führt zu einem Zeilenumbruch.

EN_XXXX

Die folgenden Benachrichtigungscodes sind Editierfeldern zugeordnet und werden dem Besitzer in *WM_COMMAND*-Botschaften übergeben, wenn eine Aktion im Editierfeld stattfindet.

EN_CHANGE Zeigt an, daß der Anwender eine Aktion durchführte, die möglicherweise den Inhalt des Textes verändert hat.

EN_ERRSPACE Zeigt an, daß kein zusätzlicher Platz im Editierfeld vorhanden ist.

EN_HSCROLL Zeigt an, daß der Anwender die horizontale Bildlaufleiste eines Editierfeldes mit der Maus angeklickt hat. Das übergeordnete Fenster wird vor dem Aktualisieren des Bildschirms benachrichtigt.

EN_KILLFOCUS Zeigt an, daß das Editierfeld den Fokus verloren hat.

EN_MAXTEXT	Zeigt an, daß die letzte Einfügung die festgelegte Anzahl von Zeichen für dieses Editierfeld überschritten hat.
EN_SETFOCUS	Zeigt an, daß das Editierfeld den Fokus erhalten hat.
EN_UPDATE	Zeigt an, daß das Editierfeld veränderten Text darstellt.
EN_VSCROLL	Zeigt an, daß der Anwender die vertikale Bildlaufleiste des Editierfeldes mit der Maus angeklickt hat. Das übergeordnete Fenster wird vor dem Aktualisieren des Bildschirms benachrichtigt.

ES_XXXX

Diese Konstanten werden verwendet, um den Stil von Editierfeldern festzulegen, wenn diese mit *CreateWindow* oder *CreateWindowEx* erzeugt werden.

ES_AUTOHSCROLL	Rollt automatisch Text um 10 Zeichen nach rechts, wenn der Anwender am Ende der Zeile ein Zeichen eintippt. Wenn der Anwender die Return -Taste drückt, rollt das Steuerelement den ganzen Text zur Nullposition zurück.
ES_AUTOVSCROLL	Rollt automatisch Text um eine Seite nach oben, wenn der Anwender in der letzten Zeile Return drückt.
ES_CENTER	Zentriert Text in einem mehrzeiligen Editierfeld. Kann nur mit *ES_MULTILINE* verwendet werden.
ES_LEFT	Richtet Text linksbündig aus. Kann nur mit *ES_MULTILINE* verwendet werden.
ES_LOWERCASE	Wandelt alle Zeichen in Kleinbuchstaben um, sowie sie im Editierfeld eingegeben werden.
ES_MULTILINE	Bestimmt ein mehrzeiliges Editierfeld (die Voreinstellung ist einzeilig). Wird der Stil *ES_AUTOVSCROLL* angegeben, zeigt das Editierfeld so viele Zeilen wie möglich und rollt vertikal, wenn der Anwender die Return -Taste drückt. Wenn *ES_AUTOVSCROLL* nicht gesetzt ist, zeigt das Editierfeld so viele Zeilen wie möglich und gibt einen Warnton aus, falls Return gedrückt wird, da keine weiteren Zeilen angezeigt werden können.

Wird der Stil *ES_AUTOHSCROLL* gesetzt, rollt das mehrzeilige Editierfeld automatisch horizontal, wenn das Caret hinter den rechten Rand des Steuerelements gelangt. Wird *ES_AUTOHSCROLL* nicht gesetzt, bricht das Steuerelement, falls nötig, automatisch Wörter auf den Anfang der nächsten Zeile um. In beiden Fällen wird eine neue Zeile auch dann begonnen, wenn Return gedrückt wird.

Die Position des Zeilenumbruchs wird von der Größe des Fensters bestimmt. Wenn die Größe des Fensters sich ändert, verändert sich die Position des Zeilenumbruchs und der Text wird neu dargestellt.

Mehrzeilige Editierfelder können Bildlaufleisten haben. Ein Editierfeld mit Bildlaufleisten bearbeitet seine eigenen Bildlaufleistenbotschaften. Editierfelder ohne Bildlaufleisten führen Bildlauf wie oben beschrieben aus und bearbeiten jede Bildlaufbotschaft, die vom übergeordneten Fenster gesendet wird.

ES_NOHIDESEL — Normalerweise macht ein Editierfeld die Auswahl unsichtbar, wenn das Steuerelement den Fokus verliert und invertiert das Auswahlfenster, wenn das Steuerelement den Fokus erhält. Die Angabe von *ES_NOHIDESEL* löscht diese voreingestellte Aktion.

ES_OEMCONVERT — Im Editierfeld eingegebener Text wird vom ANSI-Zeichensatz in den OEM-Zeichensatz und dann zurück in ANSI umgewandelt. Dies sorgt für eine genaue Konvertierung, wenn die Anwendung die Funktion *AnsiToOem* aufruft, um einen ANSI-String im Editierfeld in OEM-Zeichen umzuwandeln. Dieser Stil ist für Editierfelder hilfreich, die Dateinamen enthalten.

ES_PASSWORD — Stellt alle Zeichen als Stern (*) dar, sobald sie im Editierfeld eingegeben werden. Eine Anwendung kann die Botschaft *EM_SETPASSWORDCHAR* verwenden, um das Zeichen zu verändern, das angezeigt wird.

ES_RIGHT — Richtet Text in einem mehrzeiligen Editierfeld rechtsbündig aus.

ES_UPPERCASE — Wandelt alle Zeichen in Großbuchstaben um, sobald sie in das Editierfeld eingegeben werden.

Escape-Konstanten für serielle Schnittstellen

Diese Konstanten bestimmen den Code der Funktion, die von einer Schnittstelle ausgeführt werden soll. Sie werden in Aufrufen der Funktion *EscapeCommFunction* verwendet.

CLRDTR	Löscht das Data-Terminal-Ready-Signal (DTR).
CLRRTS	Löscht das Request-To-Sent-Signal (RTS).
RESETDEV	Wenn möglich, wird die Schnittstelle in den Ausgangszustand zurückgesetzt.
SETDTR	Sendet das Data-Terminal-Ready-Signal.
SETRTS	Sendet das Request-To-Send-Signal.
SETXOFF	Simuliert den Empfang des Zeichens XOFF.
SETXON	Simuliert den Empfang des Zeichens XON.

ETO_XXXX

Diese Optionen bestimmen die Hintergrunddarstellung durch die Funktion *ExtTextOut*.

ETO_CLIPPED	Der Text wird dem umgebenden Rechteck angepaßt.
ETO_OPAQUE	Das umgebende Rechteck wird mit der Hintergrundfarbe ausgefüllt.

EV_XXXX

Diese Konstanten dienen als Zeiger in die Ereignismaske einer Schnittstelle. Außerdem werden sie als Rückgabewert der Funktion *SetCommEventMask* verwendet.

Die folgende Aufstellung beschreibt, unter welchen Umständen die bezeichneten Maskenbits gesetzt sind.

EV_BREAK	Gesetzt, wenn eine Unterbrechung bei der Eingabe erkannt wurde.
EV_CTS	Gesetzt, wenn das Signal Clear-to-send (CTS) den Status wechselt.
EV_DSR	Gesetzt, wenn das Signal Data-set-ready (DSR) den Status wechselt.

EV_ERR	Gesetzt, wenn ein Leitungsstatusfehler auftritt. Leitungsstatusfehler sind *CE_FRAME*, *CE_OVERRUN* und *CE_RXPARITY*.
EV_PERR	Gesetzt, wenn ein Druckerfehler an einer parallelen Schnittstelle erkannt wurde. Fehler sind *CE_DNS*, *CE_IOE*, *CE_OOP* und *CE_PTO*.
EV_RING	Gesetzt, wenn ein Anrufsignal erkannt wurde.
EV_RLSD	Gesetzt, wenn das Signal Receive-line-signal-detect (RLSD) den Status ändert.
EV_RXCHAR	Gesetzt, wenn ein Zeichen empfangen und in die Warteschlange gestellt wurde.
EV_RXFLAG	Gesetzt, wenn das Ereigniszeichen empfangen und in die Empfangswarteschlange gestellt wurde. Das Ereigniszeichen wird im Gerätesteuerblock des Gerätes bestimmt.
EV_TXEMPTY	Gesetzt, wenn das letzte Zeichen in der Sendewarteschlange gesendet wurde.

FF_XXXX

Diese Konstanten bestimmen den gewünschten Zeichensatz für Schriften, die mit der Funktion *CreateFont* erzeugt werden.

FF_DECORATIVE	Ungewöhnliche Schriften, z.B. Old English.
FF_DONTCARE	Nicht wichtig oder unbekannt.
FF_MODERN	Schriften mit konstantem Zeichenabstand, mit oder ohne Serifen. Schriften mit konstantem Zeichenabstand sind normalerweise »modern«, z.B. Pica, Elite und Courier.
FF_ROMAN	Schriften mit variablem Zeichenabstand (Proportionalschriften) und mit Serifen, z.B. Times Roman, Palatino, Century Schoolbook.
FF_SCRIPT	Schriften, die wie Handschrift aussehen, z.B. Script und Cursive.
FF_SWISS	Serifenlose Schriften mit variablem Zeichenabstand (Proportionalschriften), z. B. Helvetica.

FloodFill-Flags

Diese Flags legen in der Funktion *ExtFloodFill* den Füllmodus fest.

FloodFillBorder	Der auszumalende Bereich wird von der durch *Color* festgelegten Farbe begrenzt. Diese Art des Ausmalens entspricht der Funktion *FloodFill*.
FloodFillSurface	Der auszumalende Bereich wird durch die in *Color* festgelegte Farbe definiert. Das Ausmalen wird nach außen in allen Richtungen solange fortgesetzt, wie die festgelegte Farbe angetroffen wird. Diese Art des Ausmalens ist insbesondere bei mehrfarbigen Rändern hilfreich.

FW_XXXX

Diese Konstanten bestimmen die gewünschte Schriftstärke für Schriften, die mit der Funktion *CreateFont* erzeugt werden.

Die Flags sind in aufsteigender Reihenfolge, von der dünnsten zur stärksten Schrift, aufgelistet. In der Aufzählung nebeneinander stehende Flags sind identisch.

FW_DONTCARE
FW_THIN
FW_EXTRALIGHT, FW_ULTRALIGHT
FW_LIGHT
FW_NORMAL, FW_REGULAR
FW_MEDIUM
FW_SEMIBOLD, FW_DEMIBOLD
FW_BOLD,
FW_EXTRABOLD, FW_ULTRABOLD
FW_HEAVY, FW_BLACK

GCL_XXXX

Diese Konstanten bestimmen den Byte-Offset der Fensterklassendaten, die mit der Funktion *GetClassLong* oder *SetClassLong* geliefert bzw. gesetzt werden.

GCL_MENUNAME	Zeiger auf den Menünamen (nur *SetClassLong*).
GCL_WNDPROC	Zeiger auf die Fensterfunktion.

GCW_XXXX

Diese Konstanten bestimmen den Byte-Offset der Fensterklassendaten, die mit der Funktion *GetClassWord* oder *SetClassWord* geliefert bzw. gesetzt werden.

GCW_CBCLSEXTRA	Setzt zwei neue Bytes zusätzlicher Fensterklassendaten.
GCW_CBWNDEXTRA	Setzt zwei neue Bytes zusätzlicher Fensterdaten.
GCW_HBRBACKGROUND	Setzt ein neues Handle auf einen Hintergrundpinsel.
GCW_HCURSOR	Setzt ein neues Handle auf einen Mauszeiger.
GCW_HICON	Setzt ein neues Handle auf ein Symbol.
GCW_HMODULE	Handle auf das Modul (nur *GetClassWord*).
GCW_STYLE	Setzt ein neues Stilbit für die Fensterklasse.

Gerätefähigkeits-Konstanten

Diese Konstanten entsprechen bestimmten Fähigkeiten eines Gerätekontextes. Sie werden in der Funktion *GetDeviceCaps* verwendet.

ASPECTX	Relative Breite des zum Zeichnen einer Linie verwendeten Pixel eines Geräts.
ASPECTXY	Diagonale Abmessung des zum Zeichnen einer Linie verwendeten Pixel eines Geräts.
ASPECTY	Relative Höhe des zum Zeichnen einer Linie verwendeten Pixel eines Geräts.
BITSPIXEL	Anzahl benachbarter Farb-Bits eines Pixel.
CLIPCAPS	Flag, das die Clipping-Fähigkeit des Geräts anzeigt. Die möglichen Werte sind in diesem Kapitel im Abschnitt »CP_XXXX« beschrieben.
COLORRES	Gegenwärtige Farbauflösung des Geräts in Bits je Pixel. Dieser Index ist nur dann gültig, wenn der Gerätetreiber das *RC_PALETTE*-Bit im *RASTERCAPS*-Index setzt. Er ist nur verfügbar, wenn die Versionsnummer des Treibers 3.0 oder höher ist.
CURVECAPS	Eine Bitmaske, welche die Bogenzeichnungsfähigkeit des Geräts anzeigt. Die möglichen Werte sind in diesem Kapitel im Abschnitt »CC_XXXX« beschrieben.

DRIVERVERSION	Versionsnummer, zum Beispiel 0x100 für 1.0.
HORZRES	Breite der Anzeige in Pixel.
HORZSIZE	Breite der physikalischen Anzeige in Millimetern.
LINECAPS	Eine Bitmaske, welche die Linienzeichnungsfähigkeiten des Geräts anzeigt. Die möglichen Werte sind in diesem Kapitel im Abschnitt »LC_XXXX« beschrieben.
LOGPIXELSX	Anzahl der Pixel je horizontalem Zoll.
LOGPIXELSY	Anzahl der Pixel je vertikalem Zoll.
NUMBRUSHES	Anzahl der gerätespezifischen Pinsel.
NUMCOLORS	Anzahl der Einträge in der Farbtabelle des Geräts.
NUMFONTS	Anzahl der gerätespezifischen Schriften.
NUMMARKERS	Anzahl der Markierungssymbole des Geräts.
NUMPENS	Anzahl der gerätespezifischen Stifte.
NUMRESERVED	Anzahl der reservierten Einträge in der Systempalette. Dieser Index ist nur dann gültig, wenn der Gerätetreiber das *RC_PALETTE*-Bit im *RASTERCAPS*-Index setzt. Er ist nur verfügbar, wenn die Versionsnummer des Treibers 3.0 oder höher ist.
PDEVICESIZE	Benötigte Größe des Geräte-Deskriptors.
PLANES	Anzahl der Farbebenen.
POLYGONALCAPS	Eine Bitmaske, welche die Fähigkeit des Geräts zum Zeichnen von Vielecken anzeigt. Die möglichen Werte sind in diesem Kapitel im Abschnitt »PC_XXXX« beschrieben.
RASTERCAPS	Wert, der die Scan-Fähigkeiten des Geräts anzeigt. Die möglichen Werte sind in diesem Kapitel im Abschnitt »RC_XXXX« beschrieben.
SIZEPALETTE	Anzahl der Einträge in der Systempalette. Dieser Index ist nur gültig, wenn der Gerätetreiber das *RC_PALETTE*-Bit im *RASTERCAPS*-Index setzt. Er ist nur verfügbar, wenn die Versionsnummer des Treibers 3.0 oder höher ist.
TECHNOLOGY	Gerätetechnik. Die möglichen Werte sind in diesem Kapitel im Abschnitt »DT_XXXX« beschrieben.

TEXTCAPS	Eine Bitmaske, welche die Textdarstellungsfähigkeiten des Geräts anzeigt. Die möglichen Werte sind in diesem Kapitel im Abschnitt »TC_XXXX« beschrieben.
VERTRES	Höhe der Anzeige in Scan-Zeilen.
VERTSIZE	Höhe der physikalischen Anzeige in Millimetern.

GMEM_XXXX

Diese Flags bestimmen Eigenschaften des globalen Speicherblocks, der mit der Funktion *GlobalAlloc* oder *GlobalReAlloc* erzeugt wird. Sie werden außerdem mit den Funktionen *GlobalFlags* und *GetFreeSpace* verwendet.

GMEM_DDESHARE	Reserviert gemeinsam benutzbaren Speicherplatz. Wird nur für dynamischen Datenaustausch (DDE) verwendet. Es ist jedoch zu beachten, daß Windows automatisch Speicher verwirft, der mit diesem Attribut reserviert wurde, wenn die Anwendung, die diesen Speicher belegte, die Ausführung beendet.
GMEM_DISCARDABLE	Reserviert Speicher und ordnet ihm das Attribut »discardable« zu (d.h. der Speicher kann von Windows verworfen werden). Kann nur mit *GMEM_MOVEABLE* benutzt werden.
GMEM_DISCARDED	Der Block wurde verworfen (nur *GlobalFlags*).
GMEM_FIXED	Belegt festen Speicher.
GMEM_LOCKCOUNT	Wird dieser Wert mit dem niederwertigen Byte des Rückgabewerts der Funktion *GlobalFlags* kombiniert, wird der Referenzzähler des Speicherblocks zurückgegeben (nur *GlobalFlags*).
GMEM_LOWER	Wie *GMEM_NOT_BANKED*.
GMEM_MODIFY	Die Flags des globalen Speichers werden verändert, wenn dieses Flag eingeschlossen wird (nur *GlobalReAlloc*).
GMEM_MOVEABLE	Reserviert verschiebbaren Speicher. Kann nicht mit *GMEM_FIXED* benutzt werden.
GMEM_NOCOMPACT	Verdichtet und verwirft keine anderen Speicherblöcke bei der Belegung neuen Speichers.
GMEM_NODISCARD	Verwirft keine anderen Speicherblöcke bei der Belegung neuen Speichers.

GMEM_NOT_BANKED	Belegt Speicher, der nicht in Bänke aufgeteilt ist. Kann nicht mit *GMEM_NOTIFY* benutzt werden.
GMEM_NOTIFY	Ruft die Benachrichtigungsroutine auf, falls das Speicherobjekt verworfen wird.
GMEM_SHARE	Wie *GMEM_DDESHARE*.
GMEM_ZEROINIT	Initialisiert Speicherinhalte mit Null.

GW_XXXX

Diese Konstanten bestimmen die Beziehung zwischen dem gewünschten Fenster und dem Fenster, das in Aufrufen der Funktionen *GetNextWindow* und *GetWindow* vorgegeben wird. Die Funktion *GetNextWindow* akzeptiert nur die Werte *GW_HWNDNEXT* und *GW_HWNDPREV*.

GW_CHILD	Bezeichnet das erste untergeordnete Fenster des Fensters.
GW_HWNDFIRST	Bestimmt das erste Nachbarfenster eines untergeordneten Fensters. Existiert kein solches Fenster, wird das erste Hauptfenster in der Liste angezeigt.
GW_HWNDLAST	Bestimmt das letzte Nachbarfenster für ein untergeordnetes Fenster oder das letzte Hauptfenster in der Liste.
GW_HWNDNEXT	Bestimmt das Fenster, das dem vorgegebenen Fenster in der Liste des Fenster-Managers folgt.
GW_HWNDPREV	Bestimmt das vorherige Fenster in der Liste des Fenster-Managers.
GW_OWNER	Bezeichnet den Besitzer des Fensters.

GWL_XXXX

Diese Konstanten bestimmen den Byte-Offset des Fensterattributs, das durch *GetWindowLong* oder *SetWindowLong* geliefert oder verändert werden soll.

GWL_EXSTYLE	Erweiterter Fensterstil.
GWL_STYLE	Fensterstil.
GWL_WNDPROC	Zeiger auf die Fensterfunktion.

GWW_XXXX

Diese Konstanten bestimmen den Byte-Offset der Fensterinformation, die durch die Funktionen *GetWindowWord* oder *SetWindowWord* geliefert oder verändert werden soll.

GWW_HINSTANCE — Instanz-Handle des Moduls, welches das Fenster besitzt.

GWW_HWNDPARENT — Handle des übergeordneten Fensters, falls vorhanden. Die Funktion *SetParent* ändert das übergeordnete Fenster eines untergeordneten Fensters.

Eine Anwendung sollte nicht die Funktion *SetWindowLong* aufrufen, um das übergeordnete Fenster eines untergeordneten Fensters zu ändern (nur *GetWindowWord*).

GWW_ID — Steuerelement-ID des untergeordneten Fensters.

HELP_XXXX

Diese Konstanten zeigen der Windows-Hilfefunktion die von einer Anwendung geforderte Art von Hilfe an. Sie werden in Aufrufen der Funktion *WinHelp* verwendet.

HELP_CONTEXT — Fordert Hilfe für den Kontext, der durch einen vorzeichenlosen 32-Bit-Integer in *Data* bezeichnet wird.

HELP_HELPONHELP — Fordert Hilfe für die Benutzung der Hilfeanwendung selbst. Wenn der Parameter *Command* auf *HELP_HELPONHELP* gesetzt wird, ignoriert *WinHelp* die Parameter *HelpFile* und *Data*.

HELP_INDEX — Fordert den Index der angegebenen Hilfedatei. Eine Anwendung sollte diesen Wert nur für Hilfedateien mit einem einzelnen Index und nicht zusammen mit *HELP_SETINDEX* verwenden.

HELP_KEY — Fordert Hilfe für das Schlüsselwort an, das durch einen String bezeichnet wird, auf den *Data* zeigt.

HELP_MULTIKEY — Fordert Hilfe für ein Schlüsselwort in einer veränderbaren Schlüsselworttabelle.

HELP_QUIT — Teilt der Hilfeanwendung mit, daß die angegebene Hilfedatei nicht mehr in Gebrauch ist.

HELP_SETINDEX — Setzt den durch den Parameter *Data* bestimmten Kontext als aktuellen Index für die durch den

Parameter *HelpFile* bezeichnete Hilfedatei. Dieser Index bleibt aktuell, bis der Anwender auf eine andere Hilfedatei zugreift.

Um sicherzustellen, daß der richtige Index gesetzt bleibt, sollte die Anwendung nach jedem *WinHelp*-Aufruf mit auf *HELP_CONTEXT* gesetzten *Command WinHelp* aufrufen, wobei dann *Command* auf *HELP_SETINDEX* gesetzt sein sollte und *Data* den zugehörigen Kontextbezeichner bestimmt. Eine Anwendung sollte diesen Wert nur für Hilfedateien mit mehr als einem Index benutzen.

Sie sollte diesen Wert nicht zusammen mit *HELP_INDEX* verwenden.

Hintergrund-Modi

Diese Konstanten bestimmen die Darstellung des Hintergrunds, wenn Text oder Grafik mit einem Schraffurpinsel gezeichnet wird. Sie werden mit den Funktionen *GetBkMode* und *SetBkMode* verwendet.

OPAQUE	Hintergrund wird mit der aktuellen Hintergrundfarbe ausgefüllt, bevor der Text oder die Schraffur gezeichnet wird.
TRANSPARENT	Hintergrund bleibt unberührt.

HS_XXXX

Die folgenden Konstanten bestimmen den Schraffurstil für Pinsel. Sie werden mit der Funktion *CreateHatchBrush* verwendet.

HS_BDIAGONAL	45-Grad abwärts schraffiert (von links nach rechts: \\\\\).
HS_CROSS	Horizontal und vertikal überkreuz schraffiert (+++++).
HS_DIAGCROSS	45-Grad überkreuz schraffiert (xxxxx).
HS_FDIAGONAL	45-Grad aufwärts schraffiert (von links nach rechts: /////).
HS_HORIZONTAL	Horizontal schraffiert (-----).
HS_VERTICAL	Vertikal schraffiert (|||||).

HTXXXX

Diese Codes bestimmen die Position des Mauszeigers im Verhältnis zu einem Fenster und werden in vielen Botschaften übergeben, die den Mauszeiger positionieren oder bewegen.

In Betracht kommen unter anderen die Botschaften *WM_MOUSEACTIVATE*, *WM_SETCURSOR* und *WM_NCHITTEST*.

HTBOTTOM	Auf der unteren horizontalen Grenzlinie des Fensters.
HTBOTTOMLEFT	Auf der unteren linken Ecke der Fenstergrenze.
HTBOTTOMRIGHT	Auf der unteren rechten Ecke der Fenstergrenze.
HTCAPTION	Im Titelbereich.
HTCLIENT	Im Client-Bereich.
HTERROR	Wie *HTNOWHERE*. Zusätzlich erzeugt die Funktion *DefWindowProc* einen Warnlaut, um einen Fehler anzuzeigen.
HTGROWBOX	Im Größenveränderungsfeld.
HTHSCROLL	In der horizontalen Bildlaufleiste.
HTLEFT	Auf der linken Grenzlinie des Fensters.
HTMENU	Im Menübereich.
HTNOWHERE	Auf dem Bildschirmhintergrund oder einer Trennlinie zwischen zwei Fenstern.
HTREDUCE	Im Verkleinerungsfeld.
HTRIGHT	Auf der rechten Grenzlinie des Fensters.
HTSIZE	Wie *HTGROWBOX*.
HTSYSMENU	Im Menüfeld (Schließfeld in untergeordneten Fenstern).
HTTOP	Auf der oberen horizontalen Grenzlinie des Fensters.
HTTOPLEFT	Auf der oberen linken Ecke der Fenstergrenze.
HTTOPRIGHT	Auf der oberen rechten Ecke der Fenstergrenze.
HTTRANSPARENT	In dem Fenster, das von einem anderen überdeckt wird.
HTVSCROLL	In der vertikalen Bildlaufleiste.
HTZOOM	Im Vergrößerungsfeld.

ID_XXXX

Die folgenden Konstanten werden als Rückgabewerte der Funktion *MessageBox* verwendet.

Sie bestimmen das Ergebnis des Meldungsfensters.

IDABORT	Aktionsschalter »Beenden« gedrückt.
IDCANCEL	Aktionsschalter »Abbruch« gedrückt.
IDIGNORE	Aktionsschalter »Ignorieren« gedrückt.
IDNO	Aktionsschalter »Nein« gedrückt.
IDOK	Aktionsschalter »OK« gedrückt.
IDRETRY	Aktionsschalter »Wiederholen« gedrückt.
IDYES	Aktionsschalter »Ja« gedrückt.

IDC_XXXX

Diese Bezeichner beziehen sich auf die vordefinierten Windows-Mauszeiger. Die Mauszeiger werden mit der Funktion *LoadCursor* geladen.

IDC_ARROW	Standardpfeilzeiger.
IDC_CROSS	Fadenkreuzzeiger.
IDC_IBEAM	I-Balkenzeiger in Texten.
IDC_ICON	Leeres Symbol (ineinander geschachtelte Rechtecke).
IDC_SIZE	Vierpfeilzeiger. Die Pfeile zeigen nach Norden, Osten, Süden und Westen.
IDC_SIZENESW	Doppelpfeilzeiger. Die Pfeile zeigen nach Nordost und Südwest.
IDC_SIZENS	Doppelpfeilzeiger. Die Pfeile zeigen nach Norden und Süden.
IDC_SIZENWSE	Doppelpfeilzeiger. Die Pfeile zeigen nach Nordwest und Südost.
IDC_SIZEWE	Doppelpfeilzeiger. Die Pfeile zeigen nach Westen und Osten.
IDC_UPARROW	Vertikaler Pfeilzeiger.
IDC_WAIT	Sanduhrzeiger.

IDI_XXXX

Diese Bezeichner beziehen sich auf vordefinierte Windows-Symbole. Die Symbole werden mit der Funktion *LoadIcon* geladen.

IDI_APPLICATION	Standard-Anwendungssymbol (weißes Quadrat).
IDI_ASTERISK	Ein »i« in einem Kreis (verwendet bei Informationsmeldungen).
IDI_EXCLAMATION	Ausrufezeichen (verwendet bei Warnungsmeldungen).
IDI_HAND	Symbol in Handform (verwendet bei ernsten Warnungsmeldungen).
IDI_QUESTION	Fragezeichen (verwendet bei Eingabemeldungen).

IE_XXXX

Diese Flags sind negative Werte, die einen Fehler beim Öffnen einer Schnittstelle anzeigen. Sie werden von der Funktion *OpenComm* zurückgegeben.

IE_BADID	Ungültige oder nicht unterstützte ID.
IE_BAUDRATE	Nicht unterstützte Baudrate.
IE_BYTESIZE	Ungültige Bytegröße.
IE_DEFAULT	Fehler in Standardparametern.
IE_HARDWARE	Hardware nicht vorhanden.
IE_MEMORY	Es können keine Warteschlangen eingerichtet werden.
IE_NOPEN	Gerät nicht bereit.
IE_OPEN	Gerät bereits aktiviert.

LB_XXXX

Diese Werte werden von Windows-Listenbotschaften (z.B. *LB_ADDSTRING*) zurückgegeben. Ein negativer Wert zeigt einen Fehler an.

LB_ERR	Ein Fehler ist aufgetreten. Die Aktion wurde nicht durchgeführt.
LB_ERRSPACE	Der Platz in der Liste reicht zum Durchführen der Aktion nicht aus.
LB_OKAY	Kein Fehler.

LBN_XXXX

Diese Benachrichtigungscodes gelten nur für Listen des Stils *LBS_NOTIFY*. Sie werden in *WM_COMMAND*-Botschaften übergeben und zeigen dem Besitzer der Liste eine Aktion an.

LBN_DBLCLK	Wird übergeben, wenn der Anwender einen String durch einen Doppelklick auswählt.
LBN_ERRSPACE	Zeigt an, daß im System kein Speicher mehr frei ist.
LBN_KILLFOCUS	Zeigt an, daß die Liste den Fokus verloren hat.
LBN_SELCHANGE	Wird bei Änderung der Auswahl übergeben.
LBN_SETFOCUS	Zeigt an, daß die Liste den Fokus erhalten hat.

LBS_XXXX

Diese Konstanten werden verwendet, um den Stil einer Liste festzulegen, wenn sie mit einer der Funktionen *CreateWindow* oder *CreateWindowEx* erzeugt wird.

LBS_EXTENDEDSEL	Der Anwender kann mehrere Einträge auswählen, indem er die Shift-Taste und die Maus benutzt oder eine andere spezielle Tastenkombinationen verwendet.
LBS_HASSTRINGS	Bestimmt eine besitzergezeichnete Liste, die aus Strings bestehende Einträge enthält. Die Liste behält den Speicher und die Zeiger für die Strings bei, so daß die Anwendung die Botschaft *LB_GETTEXT* verwenden kann, um den Text für einen bestimmten Eintrag abzurufen.
LBS_MULTICOLUMN	Bestimmt eine mehrspaltige Liste, die horizontal gerollt werden kann. Die Botschaft *LB_SETCOLUMNWIDTH* legt die Breite der Spalten fest.
LBS_MULTIPLESEL	Die Stringauswahl wechselt jedesmal ihren Zustand, wenn der Anwender einen String einfach oder doppelt anklickt. Jede Anzahl von Strings kann ausgewählt werden.
LBS_NOINTEGRALHEIGHT	Die Liste hat genau die Größe, die von der Anwendung angegeben wurde, als sie die Liste erzeugt hat. Normalerweise legt Windows die Größe einer Liste so fest, daß ausschließlich ganze Einträge sichtbar sind.
LBS_NOREDRAW	Die Anzeige der Liste wird nicht aktualisiert,

wenn Änderungen gemacht werden. Dieser Stil kann jederzeit geändert werden, indem eine Botschaft *WM_SETREDRAW* übergeben wird.

LBS_NOTIFY — Das übergeordnete Fenster empfängt immer dann eine Eingabebotschaft, wenn der Anwender einen String einfach oder doppelt anklickt.

LBS_OWNERDRAWFIXED — Der Besitzer zeichnet die Liste selbst. Die Einträge in der Liste haben alle die gleiche Höhe.

LBS_OWNERDRAWVARIABLE — Der Besitzer zeichnet die Liste selbst. Die Einträge in der Liste haben eine variable Höhe.

LBS_SORT — Strings in der Liste werden alphabetisch sortiert. Die Sortierreihenfolge kann in Listen des Stils *LBS_OWNERDRAWFIXED* oder *LBS_OWNERDRAWVARIABLE* abweichen.

LBS_STANDARD — Strings in der Liste werden alphabetisch sortiert, und das übergeordnete Fenster empfängt immer dann eine Eingabebotschaft, wenn der Anwender einen String einfach oder doppelt anklickt. Die Liste ist umrandet.

LBS_USETABSTOPS — Erlaubt einer Liste Tabulatorzeichen zu erkennen und zu erweitern, wenn sie ihre Strings zeichnet. Die voreingestellten Tabulatorpositionen sind 32 Dialogeinheiten. Eine Dialogeinheit ist eine horizontale oder vertikale Einheit. Eine horizontale Dialogeinheit ist gleich 1/4 der zugehörigen Basiseinheit. Die Basiseinheiten werden entsprechend der Höhe und Breite der momentanen Systemschrift berechnet. Die Funktion *GetDialogBaseUnits* liefert die gegenwärtigen Dialogbasiseinheiten in Pixel zurück.

LBS_WANTKEYBOARDINPUT — Der Besitzer der Liste erhält immer *WM_VKEYTOITEM*- oder *WM_CHARTOITEM*-Botschaften, wenn der Anwender eine Taste drückt, während die Liste den Fokus besitzt. Dies erlaubt es der Anwendung, Tastatureingaben zu bearbeiten.

LC_XXXX

Diese Konstanten repräsentieren die Linienzeichnungsfähigkeiten eines Geräts.

LC_NONE — Keine Linienzeichnungsfähigkeiten.

LC_INTERIORS — Gerät kann Ausfüllungen durchführen.

LC_MARKER	Gerät kann Markierungssymbole darstellen.
LC_POLYLINE	Gerät kann Linienzüge darstellen.
LC_POLYMARKER	Gerät kann Markierungssymbole und Linienzüge darstellen.
LC_STYLED	Gerät kann Linien verschiedener Stile darstellen.
LC_WIDE	Gerät kann breite Linien darstellen.
LC_WIDESTYLED	Gerät kann breite Linien verschiedener Stile darstellen.

LF_FACESIZE

Die Konstante *LF_FACESIZE* bestimmt die maximale Länge des Schriftbildnamens im Feld *lfFaceName* einer *LOGFONT*-Struktur. Der Wert schließt das Nullzeichen ein und beträgt 32.

LMEM_XXXX

Diese Flags bestimmen Eigenschaften des lokalen Speicherblocks, der mit den Funktionen *LocalAlloc* oder *LocalReAlloc* erzeugt wurde. *LMEM_DISCARDABLE*, *LMEM_DISCARDED* und *LMEM_LOCKCOUNT* werden mit der Funktion *LocalFlags* verwendet.

LMEM_DISCARDABLE	Der Speicher ist verwerfbar und ordnet ihm das Attribut »discardable« zu. Kann nur mit *LMEM_MOVEABLE* verwendet werden.
LMEM_DISCARDED	Block wurde verworfen (nur *LocalFlags*).
LMEM_FIXED	Der Speicherblock ist an einer festen Adresse.
LMEM_LOCKCOUNT	Wird dieser Wert mit dem niederwertigen Byte des Rückgabewerts der Funktion *LocalFlags* kombiniert, wird der Referenzzähler des Speicherblocks zurückgegeben (nur *LocalFlags*).
LMEM_MODIFY	Modifiziert das Flag *LMEM_DISCARDABLE*.
LMEM_MOVEABLE	Der Speicherblock ist verschiebbar.
LMEM_NOCOMPACT	Andere Speicherblöcke werden beim Belegen neuen Speichers weder verworfen noch verdichtet.
LMEM_NODISCARD	Andere Speicherblöcke werden beim Belegen neuen Speichers nicht verworfen.
LMEM_ZEROINIT	Initialisiert den Speicherinhalt mit Null.

LPTx

LPTx ist eine Bitmaske für das Feld *Id* einer *DCB*-Struktur. Ist die Bitmaske gesetzt, ist das Gerät an einer parallelen Schnittstelle angeschlossen.

MA_XXXX

Die Codes werden von einer *WM_MOUSEACTIVATE*-Botschaft zurückgegeben und zeigen an, ob das Fenster aktiviert und das Mausereignis verworfen werden soll.

MA_ACTIVATE	Fenster aktivieren.
MA_ACTIVATEANDEAT	Fenster aktivieren und Mausereignis verwerfen.
MA_NOACTIVATE	Fenster nicht aktivieren.

MB_XXXX

Diese Flags bestimmen die Eigenschaften eines Meldungsfensters, das mit der Funktion *MessageBox* erzeugt wurde. Sie können kombiniert werden, um den gewünschten Stil zu erzeugen.

MB_ABORT-RETRYIGNORE	Das Meldungsfenster enthält drei Schalter: *Beenden*, *Wiederholen* und *Ignorieren*.
MB_APPLMODAL	Der Anwender muß auf das Meldungsfenster reagieren, bevor er die Arbeit in dem Fenster fortsetzt, das der Parameter *Parent* bezeichnet. Der Anwender kann jedoch in Fenstern anderer Anwendungen arbeiten. *MB_APPLMODAL* ist der Standardwert, wenn weder *MB_SYSTEMMODAL* noch *MB_TASKMODAL* angegeben ist.
MB_DEFBUTTON1	Erster Schalter ist Voreinstellung. Man beachte, daß der erste Schalter immer die Voreinstellung ist, wenn nicht *MB_DEFBUTTON2* oder *MB_DEFBUTTON3* angegeben ist.
MB_DEFBUTTON2	Zweiter Schalter ist Voreinstellung.
MB_DERBUTTON3	Dritter Schalter ist Voreinstellung.
MB_ICONASTERISK	Gleichbedeutend mit *MB_ICONINFORMATION*.
MB_ICONEXCLA-MATION	Ein Ausrufungszeichensymbol erscheint im Meldungsfenster.
MB_ICONHAND	Gleichbedeutend mit *MB_ICONSTOP*.

MB_ICONINFOR-MATION	Ein Symbol, das aus einem kleinen »i« in einem Kreis besteht, erscheint im Meldungsfenster.
MB_ICONQUESTION	Ein Fragezeichensymbol erscheint im Meldungsfenster.
MB_ICONSTOP	Ein Stoppsymbol erscheint im Meldungsfenster.
MB_OK	Das Meldungsfenster enthält einen Schalter: *OK*.
MB_OKCANCEL	Das Meldungsfenster enthält zwei Schalter: *OK* und *Abbruch*.
MB_RETRYCANCEL	Das Meldungsfenster enthält zwei Schalter: *Abbruch* und *Wiederholen*.
MB_SYSTEMMODAL	Alle Anwendungen werden unterbrochen, bis der Anwender auf das Meldungsfenster reagiert. Wenn die Anwendung nicht *MB_ICONHAND* vorgibt, wird das Meldungsfenster erst modal, nachdem es erstellt wurde. Folglich empfangen das übergeordnete und andere Fenster weiterhin Botschaften, die sich aus seiner Aktivierung ergeben. Systemmodale Meldungsfenster werden verwendet, um den Benutzer von ernsten, möglicherweise schadensverursachenden Fehlern zu benachrichtigen, die sofortige Aufmerksamkeit erfordern (z.B. Speicherüberlauf).
MB_TASKMODAL	Wie *MB_APPLMODAL*, mit der Ausnahme, daß alle Hauptfenster, die zur aktuellen Task gehören, gesperrt werden, wenn der Parameter *Parent* 0 ist. Dieses Flag sollte dann verwendet werden, wenn die aufrufende Anwendung oder Bibliothek kein verfügbares Fenster-Handle hat, aber eines benötigt, um Eingaben in andere Fenster der aktuellen Anwendung zu verhindern, ohne andere Anwendungen zu unterbrechen.
MB_YESNO	Meldungsfenster enthält zwei Schalter: *Ja* und *Nein*.
MB_YESNOCANCEL	Meldungsfenster enthält drei Schalter: *Ja*, *Nein* und *Abbruch*.

Für bestimmte Gruppen von *MB_*KONSTANTEN sind Bitmasken definiert.

Maske	Enthaltene Konstanten
MB_DEFMASK	*MB_DEFBUTTON1, MB_DEFBUTTON2, MB_DEFBUTTON3*
MB_ICONMASK	*MB_ICONASTERISK, MB_ICONEXCLAMATION, MB_ICONHAND, MB_ICONINFORMATION, MB_ICONQUESTION, MB_ICONSTOP*

Maske	Enthaltene Konstanten
MB_MODEMASK	MB_APPLMODAL, MB_SYSTEMMODAL, MB_TASKMODAL
MB_TYPEMASK	MB_ABORTRETRYIGNORE, MB_OK, MB_OKCANCEL, MB_RETRYCANCEL, MB_YESNO, MB_YESNOCANCEL

META_XXXX

Diese Konstanten entsprechen einzelnen Funktionen der grafischen Geräteschnittstelle (GDI). Der Wert von *META_ARC* ist also der numerische Index der GDI-Funktion *Arc*.

Die Indexwerte dieser Funktionen und die entsprechenden Konstanten können in einem Metafile gespeichert werden.

Dieses Metafile ist eine Liste von GDI-Befehlen, die von einer Anwendung zur Erzeugung grafischer Ausgaben abgespielt werden kann.

META_ANIMATEPALETTE
META_ARC
META_BITBLT
META_CHORD
META_CREATEBITMAP
META_CREATEBITMAPINDIRECT
META_CREATEBRUSH
META_CREATEBRUSHINDIRECT
META_CREATEFONTINDIRECT
META_CREATEPALETTE
META_CREATEPATTERNBRUSH
META_CREATEPENINDIRECT
META_CREATEREGION
META_DELETEOBJECT
META_DIBBITBLT
META_DIBCREATEPATTERNBRUSH
META_DIBSTRETCHBLT
META_DRAWTEXT
META_ELLIPSE
META_ESCAPE
META_EXCLUDECLIPRECT
META_EXTTEXTOUT
META_FILLREGION
META_FLOODFILL
META_FRAMEREGION
META_INTERSECTCLIPRECT
META_INVERTREGION
META_LINETO
META_MOVETO
META_OFFSETCLIPRGN
META_OFFSETVIEWPORTORG
META_OFFSETWINDOWORG
META_PAINTREGION
META_PATBLT
META_PIE
META_POLYGON
META_POLYLINE
META_POLYPOLYGON
META_REALIZEPALETTE
META_RECTANGLE
META_RESIZEPALETTE
META_RESTOREDC
META_ROUNDRECT
META_SAVEDC
META_SCALEVIEWPORTEXT
META_SCALEWINDOWEXT
META_SELECTCLIPREGION
META_SELECTOBJECT
META_SELECTPALETTE
META_SETBKCOLOR
META_SETBKMODE
META_SETDIBTODEV
META_SETMAPMODE
META_SETMAPPERFLAGS
META_SETPALENTRIES
META_SETPIXEL
META_SETPOLYFILLMODE
META_SETRELABS
META_SETROP2
META_SETSTRETCHBLTMODE

META_SETTEXTALIGN
META_SETTEXTCHAREXTRA
META_SETTEXTCOLOR
META_SETTEXTJUSTIFICATION
META_SETVIEWPORTEXT
META_SETVIEWPORTORG
META_SETWINDOWEXT
META_SETWINDOWORG
META_STRETCHBLT
META_TEXTOUT

MF_XXXX

Die folgenden Konstanten werden als Flags in vielen Menü-Funktionen und in der Botschaft *WM_MENUSELECT* verwendet.

MF_BITMAP Verwendet ein Bitmap als Menüoption.

MF_BYCOMMAND Bestimmt die Befehls-ID der existierenden Menüoption.

MF_BYPOSITION Bestimmt die Position einer existierenden Menüoption. Die erste Option im Menü liegt an Position 0.

MF_CHECKED Setzt eine Auswahlmarkierung neben die Menüoption.

Wenn die Anwendung Bitmaps für Auswahlmarkierungen zur Verfügung gestellt hat (siehe auch *SetMenuItemBitmaps*), bewirkt das Setzen dieses Flags die Anzeige des Bitmaps »Ausgewählt« neben der Menüoption.

MF_DISABLED Sperrt die Menüoption, so daß sie nicht ausgewählt werden kann, versetzt sie aber nicht in den nichtaktivierbaren Zustand.

MF_ENABLED Schaltet die Menüoption ein, so daß sie ausgewählt werden kann und stellt sie aus dem nichtaktivierbaren Zustand wieder her.

MF_GRAYED Sperrt die Menüoption, so daß sie nicht ausgewählt werden kann und versetzt sie in den nichtaktivierbaren Zustand.

MF_HELP Zeigt an, daß der Eintrag ein Hilfe-Menüeintrag ist.

MF_HILITE Hebt einen Menüeintrag hervor.

MF_MENUBARBREAK Gleichbedeutend mit *MF_MENUBREAK* mit der Ausnahme, daß bei Pop-Up-Menüs die neue Spalte von der alten durch eine vertikale Linie getrennt wird.

MF_MENUBREAK Setzt die Menüoption auf eine neue Zeile für statische Menüleistenelemente. Bei Pop-Up-

Menüs setzt dieses Flag das Element in eine neue Spalte, ohne Trennungslinie zwischen den Spalten.

MF_MOUSESELECT — Zeigt an, daß der Eintrag mit der Maus ausgewählt wurde.

MF_OWNERDRAW — Bestimmt, daß die Menüoption eine besitzergezeichnete Option ist. Das Fenster, das das Menü besitzt, empfängt eine Botschaft *WM_MEASUREITEM*, wenn das Menü das erste Mal dargestellt wird, um Höhe und Breite der Menüoption abzurufen.

Die Botschaft *WM_DRAWITEM* wird immer dann gesendet, wenn die Anwendung das Erscheinungsbild der Menüoption aktualisieren muß. Diese Einstellung des Parameters gilt nicht für Hauptmenüoptionen.

MF_POPUP — Bestimmt, daß der Option ein Pop-up-Menü zugeordnet ist.

Dieses Flag sollte verwendet werden, um entweder ein Haupt-Pop-Up-Menü einzufügen oder ein hierarchisches Pop-Up-Menü an eine Pop-Up-Menüoption anzufügen.

MF_SEPARATOR — Zeichnet eine horizontale Trennungslinie.

Dieses Flag ist nur in einem Pop-Up-Menü zu verwenden. Diese Linie kann nicht in den nichtaktivierbaren Zustand versetzt, gesperrt oder hervorgehoben werden.

MF_STRING — Bestimmt, daß die Menüoption ein String ist.

MF_SYSMENU — Zeigt an, daß sich der Eintrag im Systemmenü befindet.

MF_UNCHECKED — Setzt keine Auswahlmarkierung neben die Menüoption. Keine Auswahl ist die Voreinstellung, wenn weder *MF_CHECKED* noch *MF_UNCHECKED* gesetzt ist.

Wenn die Anwendung Bitmaps für Auswahlmarkierungen zur Verfügung gestellt hat (siehe *SetMenuItemBitmaps*), bewirkt das Setzen dieses Flags, daß das Bitmap »Nicht ausgewählt« neben der Menüoption dargestellt wird.

MF_UNHILITE — Entfernt die Hervorhebung des Menüeintrags.

MK_XXXX

Diese Masken werden kombiniert und in Mausklickbotschaften übergeben, um den Status bestimmter Tasten und Maustasten zu ermitteln.

MK_CONTROL	Beim Drücken der Ctrl-Taste gesetzt.
MK_LBUTTON	Beim Drücken der linken Maustaste gesetzt.
MK_MBUTTON	Beim Drücken der mittleren Maustaste gesetzt.
MK_RBUTTON	Beim Drücken der rechten Maustaste gesetzt.
MK_SHIFT	Beim Drücken der Shift -Taste gesetzt.

MM_XXXX

Diese Konstanten bestimmen den Abbildungsmodus des Gerätekontextes. Dieser Modus bestimmt die Konvertierung virtueller Einheiten in Geräteeinheiten und die Ausrichtung der Achsen.

Die Konstanten werden in den Funktionen *GetMapMode* und *SetMapMode* verwendet.

MM_ANISOTROPIC	Für Anwendungen, die eine virtuelle Einheit auf eine beliebige physikalische Einheit abbilden. Die x- und y-Achsen können beliebig skaliert werden.
MM_HIENGLISH	Für Anwendungen, die eine virtuelle Einheit auf 0.001 Zoll abbilden. Positive y-Richtung nach oben.
MM_HIMETRIC	Für Anwendungen, die eine virtuelle Einheit auf 0.01 Millimeter abbilden. Positive y-Richtung nach oben.
MM_ISOTROPIC	Für Anwendungen, die eine virtuelle Einheit auf eine beliebige physikalische Einheit abbilden. Eine x-Achseneinheit ist gleich einer y-Achseneinheit.
MM_LOENGLISH	Für Anwendungen, die eine virtuelle Einheit auf 0.01 Zoll abbilden. Positive y-Richtung nach oben.
MM_LOMETRIC	Für Anwendungen, die eine virtuelle Einheit auf 0.1 Millimeter abbilden. Positive y-Richtung nach oben.
MM_TEXT	Für Anwendungen, die eine virtuelle Einheit auf ein Pixel abbilden. Positive y-Richtung nach unten.

MM_TWIPS Für Anwendungen, die eine virtuelle Einheit auf 1/1440 Zoll abbilden. Positive y-Richtung nach oben.

MSGF_XXXX

Diese Codes werden im Parameter *Code* der beiden Botschaftsfilterfunktionen *WH_MSGFILTER* und *WH_SYSMSGFILTER* übergeben. Sie bestimmen den Typ der zu bearbeitenden Botschaften.

	Zu bearbeitende Botschaften *WH_MSGFILTER*	Konstante *WH_SYSMSGFILTER*
MSGF_DIALOGBOX	Meldungs- und Dialogfenster	Nur Dialogfenster
MSGF_MENU	Tastatur- und Mausbotschaften	Tastatur- und Mausbotschaften
MSGF_MESSAGEBOX		Nur Meldungsfensterbotschaften

OBJ_XXXX

Diese Konstanten bestimmen den Typ des GDI-Objekts (Stift oder Pinsel), das in einem Aufruf der Funktion *EnumObjects* aufgezählt werden soll.

Sie sollten als Parameter *ObjectType* der Funktion *EnumObjects* übergeben werden.

OBJ_PEN Zählt Stifte auf.

OBJ_BRUSH Zählt Pinsel auf.

OBM_XXXX

Diese Konstanten bestimmen die Bitmaps, die unter Windows für die Verwendung durch das System vordefiniert sind. Programmierer können mit der Funktion *LoadBitmap* auf diese Bitmaps zugreifen. Konstanten, die mit *OBM_OLD_* beginnen, beziehen sich auf Bitmaps, die in Windows-Versionen vor 3.0 verwendet wurden.

OBM_BTNCORNERS
OBM_BTSIZE
OBM_CHECK
OBM_CHECKBOXES
OBM_CLOSE
OBM_COMBO
OBM_DNARROW
OBM_DNARROWD
OBM_LFARROW
OBM_LFARROWD
OBM_MNARROW
OBM_OLD_CLOSE
OBM_OLD_DNARROW
OBM_OLD_LFARROW

OBM_OLD_REDUCE
OBM_OLD_RESTORE
OBM_OLD_RGARROW
OBM_OLD_UPARROW
OBM_OLD_ZOOM
OBM_REDUCE
OBM_REDUCED
OBM_RESTORE
OBM_RESTORED
OBM_RGARROW
OBM_RGARROWD
OBM_SIZE
OBM_UPARROW
OBM_UPARROWD
OBM_ZOOM
OBM_ZOOMD

ODA_XXXX

Diese Konstanten definieren die Aktion, die für ein besitzergezeichnetes Steuerelement durchgeführt werden soll. Sie werden im Feld *itemAction* der Struktur *DRAWITEMSTRUCT* verwendet. Das Feld kann auch eine Kombination dieser Konstanten aufnehmen.

ODA_DRAWENTIRE	Das ganze Steuerelement muß gezeichnet werden.
ODA_FOCUS	Der Fokus wurde empfangen oder abgegeben.
ODA_SELECT	Der Auswahlstatus wurde verändert.

ODS_XXXX

Diese Konstanten definieren den Status eines besitzergezeichneten Steuerelements nach seiner Zeichnung. Sie werden im Feld *itemState* der Struktur *DRAWITEMSTRUCT* verwendet. Das Feld kann auch eine Kombination dieser Werte aufnehmen.

ODS_CHECKED	Menüeintrag markieren.
ODS_DISABLED	Menüeintrag sperren.
ODS_FOCUS	Der Menüeintrag erhält den Fokus.
ODS_GRAYED	Menüeintrag wird in den nichtaktivierbaren Zustand versetzt.
ODS_SELECTED	Menüeintrag auswählen.

ODT_XXXX

Diese Konstanten bestimmen den Typ des besitzergezeichneten Steuerelements. Sie werden im Feld *CtlType* der Strukturen *COMPAREITEMSTRUCT*, *DRAWITEMSTRUCT* und *MEASUREITEMSTRUCT* verwendet.

ODT_MENU	Besitzergezeichnetes Menü.
ODT_LISTBOX	Besitzergezeichnete Liste.
ODT_COMBOBOX	Besitzergezeichnetes Kombinationsfenster.
ODT_BUTTON	Besitzergezeichneter Schalter.

OF_XXXX

Diese Konstanten werden kombiniert und in Aufrufen der Funktionen *OpenFile* oder *_lopen* übergeben. Sie bestimmen die Art der Dateiöffnung.

OF_CANCEL	Fügt einen Aktionsschalter für Abbruch in das Dialogfenster *OF_PROMPT* ein. Die Betätigung des Abbruchschalters weist *OpenFile* an, die Fehlermeldung `Datei nicht gefunden` zurückzugeben (nur *OpenFile*).
OF_CREATE	Weist *OpenFile* an, eine neue Datei zu erstellen. Falls die Datei schon vorhanden ist, wird sie auf die Länge 0 gekürzt (nur *OpenFile*).
OF_DELETE	Löscht die Datei (nur *OpenFile*).
OF_EXIST	Öffnet die Datei und schließt sie wieder. Wird für eine Prüfung auf die Existenz einer Datei verwendet (nur *OpenFile*).
OF_PARSE	Füllt die Datenstruktur *OFSTRUCT*, die im Parameter *ReOpenBuff* übergeben wird. Weitere Aktionen finden nicht statt.
OF_PROMPT	Zeigt ein Dialogfenster an, wenn die angesprochene Datei nicht existiert. Das Dialogfenster informiert den Benutzer, daß Windows die Datei nicht finden kann und fordert den Benutzer auf, eine Diskette mit der Datei in Laufwerk A: einzulegen (nur *OpenFile*).
OF_READ	Öffnet die Datei nur zum Lesen.
OF_READWRITE	Öffnet die Datei zum Lesen und Schreiben.
OF_REOPEN	Öffnet die Datei unter Verwendung der Informationen in der durch *ReOpenBuff* bezeichneten Struktur *OFSTRUCT* (nur *OpenFile*).
OF_SHARE_COMPAT	Öffnet eine Datei im Kompatibilitätsmodus und ermöglicht dadurch beliebig häufiges Öffnen der Datei. Die Funktion kann nicht ausgeführt werden, wenn die Datei in einem anderen Modus für gemeinsame Nutzung (*Share*) geöffnet wurde.

OF_SHARE_DENY_NONE	Öffnet die Datei, ohne anderen Prozessen den Lese- oder Schreibzugriff auf diese Datei zu verweigern. Die Funktion kann nicht ausgeführt werden, wenn die Datei von einem anderen Prozeß im Kompatibilitätsmodus geöffnet wurde.
OF_SHARE_DENY_READ	Öffnet die Datei und verweigert anderen Prozessen den Lesezugriff auf die Datei. Die Funktion kann nicht ausgeführt werden, wenn die Datei im Kompatibilitätsmodus oder für Lesezugriffe von einem anderen Prozeß geöffnet wurde.
OF_SHARE_DENY_WRITE	Öffnet die Datei und verweigert anderen Prozessen Schreibzugriffe auf die Datei. Die Funktion kann nicht ausgeführt werden, wenn die Datei im Kompatibilitätsmodus oder für Schreibzugriff von einem anderen Prozeß geöffnet wurde.
OF_SHARE_EXCLUSIVE	Öffnet die Datei im Exklusivmodus und verweigert anderen Prozessen sowohl Lese- als auch Schreibzugriffe auf die Datei. Die Funktion kann nicht ausgeführt werden, wenn die Datei in irgendeinem anderen Modus für Lese- oder Schreibzugriffe geöffnet wurde. Das gilt auch für den aktuellen Prozeß.
OF_VERIFY	Überprüft, ob Datum und Zeit der Datei dieselben sind, wie beim vorherigen Öffnen. Dieser Wert dient zur zusätzlichen Überprüfung von schreibgeschützten Dateien (nur *OpenFile*).
OF_WRITE	Öffnet die Datei nur zum Schreiben.

OUT_XXXX

Diese Konstanten bestimmen die Ausgabepräzision von Schriften, die mit der Funktion *CreateFont* erzeugt werden.

OUT_CHARACTER_PRECIS	*OUT_STRING_PRECIS*
OUT_DEFAULT_PRECIS	*OUT_STROKE_PRECIS*

PC_XXXX

Diese Flags bestimmen Informationen über eine Palette. Sie werden im Feld *peFlags* der Struktur *PALETTEENTRY* verwendet und in Aufrufen einer der Funktionen *AnimatePalette*, *GetPaletteEntries*, *GetSystemPaletteEntries* und *SetPaletteEntries* übergeben.

PC_EXPLICIT Bestimmt, daß das niederwertige WORD des virtuellen Paletteneintrags einen Hardware-Palettenindex beschreibt. Dieses Flag erlaubt einer Anwendung, den Inhalt der Palette eines Anzeigegeräts anzuzeigen.

PC_NOCOLLAPSE Bestimmt, daß die Farbe in einem ungenutzten Eintrag in der Systempalette abgelegt wird, anstelle der Anpassung an einen existierenden Farbeintrag in der Systempalette. Sind keine ungenutzten Einträge in der Systempalette vorhanden, wird die Farbe normal angepaßt.

Ist diese Farbe erst in der Systempalette eingetragen, können Farben in anderen virtuellen Paletten dieser Farbe angepaßt werden.

PC_RESERVED Bestimmt, daß der virtuelle Paletteneintrag für Palettenanimation genutzt wird. Dadurch wird verhindert, daß andere Fenster die Farben an diesen Paletteneintrag anpassen, da sich die Farbe zeitweilig ändert. Wenn ein ungenutzter Systempaletteneintrag verfügbar ist, wird diese Farbe in den Eintrag plaziert, andernfalls steht diese Farbe für Animation nicht zur Verfügung.

Diese Konstanten repräsentieren die Polygonzeichnungsfähigkeiten eines Geräts.

PC_INTERIORS Gerät kann Ausfüllungen durchführen.

PC_NONE Gerät unterstützt die Darstellung von Polygonen nicht.

PC_POLYGON Gerät kann Polygone darstellen.

PC_RECTANGLE Gerät kann Rechtecke darstellen.

PC_SCANLINE Gerät kann Rasterzeilen darstellen.

PC_STYLED Gerät kann Rahmen verschiedener Stile darstellen.

PC_TRAPEZOID Gerät kann Trapeze darstellen.

PC_WIDE Gerät kann breite Rahmen darstellen.

PC_WIDESTYLED Gerät kann breite Rahmen verschiedener Stile darstellen.

PC_WINDPOLYGON Gerät kann überschneidende Polygone darstellen.

PM_XXXX

Diese Optionen bestimmen den Umgang mit Botschaften, die von der Funktion *PeekMessage* verarbeitet werden.

PM_NOREMOVE	Botschaften werden nach der Bearbeitung durch *PeekMessage* nicht aus der Warteschlange gelöscht.
PM_NOYIELD	Läßt andere Tasks die aktuelle nicht unterbrechen.
PM_REMOVE	Botschaften werden nach der Bearbeitung durch *PeekMessage* aus der Warteschlange gelöscht.

Polygonfüll-Modi

Diese Konstanten bestimmen die Methode, nach der Polygone mit mehr als einer assoziierten Region gefüllt werden. Sie werden in Aufrufen der Funktionen *CreatePolygonRgn*, *CreatePolyPolygonRgn*, *GetPolyFillMode* und *SetPolyFillMode* verwendet.

ALTERNATE	Regionen werden alternierend ausgefüllt.
WINDING	Jede Region wird gefüllt.

PR_JOBSTATUS

PR_JOBSTATUS wird als Parameter *wParam* der Botschaft *WM_SPOOLERSTATUS* übergeben.

PROC_XXXX

Diese Konstanten werden als Parameter *ProcName* der Funktion *GetProcAdress* übergeben, wenn das Modul-Handle im Parameter *Module* einen Gerätetreiber bezeichnet.

PROC_EXTDEVICEMODE
PROC_DEVICECAPABILITIES
PROC_OLDDEVICEMODE

PS_XXXX

Die folgenden Konstanten legen Stiftstile fest. Sie werden in der Funktion *CreatePen* verwendet.

PS_SOLID	_____
PS_DASH	- - - - -
PS_DOT	. . . . .
PS_DASHDOT	- · - · - · - · -
PS_DASHDOTDOT	- ·· - ·· -
PS_NULL	Unsichtbar
PS_INSIDEFRAME	Wird zum Zeichnen des inneren Rahmens von Polygonen und Linienzügen verwendet.

R2_XXXX

Diese Konstanten bestimmen den Zeichenmodus, um die Farben des aktuellen Stifts mit den existierenden Farben zu kombinieren. Sie werden als Parameter in den Funktionen *GetROP2* und *SetROP2* übergeben.

R2_BLACK	Pixel ist immer schwarz.
R2_COPYPEN	Pixel hat die Stiftfarbe.
R2_MASKNOTPEN	Pixel ist eine Kombination der Farben, die dem Bildschirm und der inversen Stiftfarbe gemeinsam sind.
R2_MASKPEN	Pixel ist eine Kombination der Farben, die dem Stift und dem Bildschirm gemeinsam sind.
R2_MASKPENNOT	Pixel ist eine Kombination der Farben, die dem Stift und der inversen Bildschirmfarbe gemeinsam sind.
R2_MERGENOTPEN	Pixel ist eine Kombination der Bildschirmfarbe und der inversen Stiftfarbe.
R2_MERGEPEN	Pixel ist eine Kombination der Stiftfarbe und der Bildschirmfarbe.
R2_MERGEPENNOT	Pixel ist eine Kombination der Stiftfarbe und der inversen Bildschirmfarbe.
R2_NOP	Pixel bleibt unverändert.
R2_NOT	Pixel ist invers zur Bildschirmfarbe.
R2_NOTCOPYPEN	Pixel hat die inverse Stiftfarbe.

R2_NOTMASKPEN	Pixel ist die Invertierung der Farbe *R2_MASKPEN*.
R2_NOTMERGEPEN	Pixel ist die Invertierung der Farbe *R2_MERGEPEN*.
R2_NOTXORPEN	Pixel ist die Invertierung der Farbe *R2_XORPEN*.
R2_WHITE	Pixel ist immer weiß.
R2_XORPEN	Pixel ist eine Kombination der Farben im Stift und auf dem Bildschirm, die nicht in beiden gleichzeitig vorkommen.

RC_XXXX

Diese Konstanten repräsentieren die Rasterfähigkeiten eines Geräts.

RC_BANDING	Benötigt Frequenzunterstützung.
RC_BIGFONT	Unterstützt mehr als 64 KByte große Schriften.
RC_BITBLT	Fähig zur Bitmap-Übertragung.
RC_BITMAP64	Fähig zur Unterstützung von mehr als 64 KByte großen Bitmaps.
RC_DI_BITMAP	Fähig zur Unterstützung von *SetDIBits* und *GetDIBits*.
RC_DIBTODEV	Fähig zur Unterstützung von *SetDIBitsToDevice*.
RC_FLOODFILL	Fähig zum Ausmalen, unterstützt die Funktion *FloodFill*.
RC_GDI20_OUTPUT	Fähig zur Unterstützung von Merkmalen der Version Windows 2.0.
RC_PALETTE	Auf Farbpalette aufbauendes Gerät.
RC_SCALING	Fähigkeit zum Skalieren.
RC_STRETCHBLT	Fähig zur Durchführung von *StretchBlt*.
RC_STRETCHDIB	Fähig zur Durchführung von *StretchDIBits*.

Region-Flags

Diese Flags bestimmen den Typ einer grafischen Region. Sie werden als Rückgabewerte der Funktionen *CombineRgn*, *ExcludeClipRect*, *ExcludeUpdateRgn*, *GetClipBox*, *GetRgnBox*, *GetUpdateRgn*, *OffsetClipRgn*, *OffsetRgn* und *SelectClipRgn* verwendet.

COMPLEXREGION	Die neue Region hat überlappende Ränder.
ERROR	Keine neue Region erzeugt.
NULLREGION	Die neue Region ist leer.
SIMPLEREGION	Die neue Region hat keine überlappenden Ränder.

RGN_XXXX

Diese Flags bestimmen die Kombinationsmethode, die von der Funktion *CombineRgn* zur Kombination zweier Regionen verwendet wird.

RGN_AND	Verwendet überlappende Bereiche beider Regionen (Schnittpunkt).
RGN_COPY	Erzeugt eine Kopie von Region 1 (wird durch *SrcRgn1* bezeichnet).
RGN_DIFF	Speichert die Bereiche der durch den Parameter *SrcRgn1* bezeichneten Region 1, die nicht Teil der durch *SrcRgn2* bezeichneten Region 2 sind.
RGN_OR	Vereinigt beide Regionen.
RGN_XOR	Vereinigt die sich nicht überlappenden Teile beider Regionen.

RT_XXXX

Diese Konstanten repräsentieren die verfügbaren Typen von Windows-Ressourcen. Sie werden als Parameter der Funktion *FindResource* verwendet.

RT_ACCELERATOR	Akzelerator-Tabelle.
RT_BITMAP	Bitmap-Ressource.
RT_CURSOR	Mauszeigerressource.
RT_DIALOG	Dialogfenstervorlage.
RT_FONT	Schriftressource.
RT_FONTDIR	Schriftenverzeichnisressource.
RT_ICON	Symbolressource.
RT_MENU	Menüressource.
RT_RCDATA	Anwenderdefinierte Ressource (Rohdaten).
RT_STRING	String-Ressource.

S_XXXX

Diese Konstanten werden in verschiedenen Sound-Funktionen verwendet. Sie werden entsprechend der Funktionen aufgelistet, in denen sie verwendet werden. Folgende negativen Fehlerwerte werden von Sound-Funktionen zurückgegeben.

S_SERBDNT	Ungültige Note.
S_SERDCC	Ungültiger Notenzähler.
S_SERDDR	Ungültige Dauer.
S_SERDFQ	Ungültige Frequenz.
S_SERDLN	Ungültige Notenlänge.
S_SERDMD	Ungültiger Modus.
S_SERDPT	Ungültige Tonhöhe.
S_SERDSH	Ungültige Hüllkurve.
S_SERDSR	Ungültige Quelle.
S_SERDST	Ungültiger Status.
S_SERDTP	Ungültiges Tempo.
S_SERDVL	Ungültige Lautstärke.
S_SERDVNA	Gerät ist nicht verfügbar.
S_SERMACT	Musik ist aktiv.
S_SEROFM	Speicherüberlauf.
S_SERQFUL	Warteschlange ist voll.

SetSoundNoise

Diese Konstanten bestimmen die Tonquelle. *N* ist ein Wert zur Ermittlung der Zielfrequenz.

S_PERIOD512	Quellfrequenz ist N/512 (hoch); dunkler Ton.
S_PERIOD1024	Quellfrequenz ist N/1024.
S_PERIOD2048	Quellfrequenz ist N/2048 (tief); heller Ton.
S_PERIODVOICE	Quellfrequenz von Voice-Kanal 3.
S_WHITE512	Quellfrequenz ist N/512 (hoch); dunkler Ton.
S_WHITE1024	Quellfrequenz ist N/1024.
S_WHITE2048	Quellfrequenz ist N/2048 (tief); heller Ton.
S_WHITEVOICE	Quellfrequenz von Voice-Kanal 3.

SetVoiceAccent
Diese Konstanten bestimmen die Phrasierung.

S_LEGATO	Der Ton wird für die ganze Dauer gehalten und bei Beginn des nächsten Tons ausgeblendet.
S_NORMAL	Der Ton wird für die ganze Dauer gehalten und bricht ab, bevor der nächste Ton beginnt.
S_STACCATO	Der Ton wird nur für einen Teil der Dauer gehalten und weist einen deutlichen Abstand zwischen ihm und dem nächsten Ton auf.

WaitSoundState
Diese Konstanten bestimmen den Zustand der Klangwarteschlangen.

S_ALLTHRESHOLD	Alle Klangwarteschlangen haben den Schwellenwert erreicht.
S_QUEUEEMPTY	Alle Klangwarteschlangen sind leer und die Klangtreiber sind ausgeschaltet.
S_THRESHOLD	Eine Klangwarteschlange hat den Schwellenwert erreicht und gibt ihren Inhalt aus.

SB_XXXX

Diese Konstanten bezeichnen Bildlaufleistenereignisse und werden als Parameter in den Botschaften *WM_HSCROLL*, *WM_HSCROLLCLIPBOARD*, *WM_VSCROLL* und *WM_VSCROLLCLIPBOARD* übergeben.

SB_BOTTOM	Bildlauf nach unten oder nach rechts.
SB_ENDSCROLL	Ende des Bildlaufs.
SB_LINEDOWN	Bildlauf eine Zeile nach unten.
SB_LINEUP	Bildlauf eine Zeile nach oben.
SB_PAGEDOWN	Bildlauf eine Seite nach unten.
SB_PAGEUP	Bildlauf eine Seite nach oben.
SB_THUMBPOSITION	Bildlauf zu einer absoluten Position. Die aktuelle Position wird in *LOWORD(lParam)* bereitgestellt.
SB_THUMBTRACK	Mauszug zur festgelegten Position. Die aktuelle Position wird in *LOWORD(lParam)* bereitgestellt.
SB_TOP	Bildlauf nach oben oder nach links.

Diese Konstanten bezeichnen die Bildlaufleiste, die durch die Funktionen bzw. Prozeduren *GetScrollPos*, *GetScrollRange*, *SetScrollPos*, *SetScrollRange* und *ShowScrollBar* bestimmt wird.

SB_BOTH	Bestimmt die horizontale und die vertikale Bildlaufleiste des Fensters.
SB_CTL	Bestimmt eine selbständige Bildlaufleiste.
SB_HORZ	Bestimmt die horizontale Bildlaufleiste des Fensters.
SB_VERT	Bestimmt die vertikale Bildlaufleiste des Fensters.

SBS_XXXX

Diese Konstanten werden verwendet, um den Bildlaufleistenstil festzulegen, wenn eine Bildlaufleiste durch die Funktionen *CreateWindow* oder *CreateWindowEx* erzeugt wird.

SBS_BOTTOMALGIN	Wird mit dem Stil *SBS_HORZ* verwendet. Der untere Rand der Bildlaufleiste wird am unteren Rand des Rechtecks ausgerichtet, das durch die in der Funktion *CreateWindow* angegebenen Parameter *X*, *Y*, *Width* und *Height* festgelegt wird. Die Bildlaufleiste hat die voreingestellte Höhe für Systembildlaufleisten.
SBS_HORZ	Bestimmt eine horizontale Bildlaufleiste. Sind weder der Stil *SBS_BOTTOMALIGN* noch der Stil *SBS_TOPALIGN* gesetzt, hat die Bildlaufleiste die mit der Funktion *CreateWindow* gesetzte Höhe, Breite und Position.
SBS_LEFTALGIN	Wird mit dem Stil *SBS_VERT* verwendet. Der linke Rand der Bildlaufleiste wird am linken Rand des Rechtecks ausgerichtet, das durch die mit der Funktion *CreateWindow* gesetzten Parameter *X*, *Y*, *Width* und *Height* festgelegt wird. Die Bildlaufleiste hat die voreingestellte Breite für Systembildlaufleisten.
SBS_RIGHTALGIN	Wird mit dem Stil *SBS_VERT* benutzt. Der rechte Rand der Bildlaufleiste wird am rechten Rand des Rechtecks ausgerichtet, das durch die in der Funktion *CreateWindow* angegebenen Parameter *X*, *Y*, *Width* und *Height* festgelegt wird. Die Bildlaufleiste hat die voreingestellte Breite für Systembildlaufleisten.
SBS_SIZEBOX	Bestimmt ein Größenveränderungsfeld. Wenn weder der Stil *SBS_SIZEBOXBOTTOMRIGHTALIGN* noch der Stil *SBS_SIZEBOXTOPLEFTALIGN*

gesetzt sind, hat das Größenveränderungsfeld die in der Funktion *CreateWindow* angegebene Höhe, Breite und Position.

SBS_SIZEBOXBOTTOM-RIGHTALGIN

Wird mit dem Stil *SBS_SIZEBOX* verwendet. Die untere rechte Ecke des Größenveränderungsfeldes wird an der unteren rechten Ecke des Rechtecks ausgerichtet, das durch die in der Funktion *CreateWindow* angegebenen Parameter *X*, *Y*, *Width* und *Height* festgelegt wird.

Das Größenveränderungsfeld hat die voreingestellte Größe des Systemgrößenveränderungsfeldes.

SBS_SIZEBOXTOP-LEFTALGIN

Wird mit dem Stil *SBS_SIZEBOX* verwendet. Die obere linke Ecke des Größenveränderungsfeldes wird an der oberen linken Ecke des Rechtecks ausgerichtet, das durch die in der Funktion *CreateWindow* angegebenen Parameter *X*, *Y*, *Width* und *Height* festgelegt wird. Das Größenveränderungsfeld hat die voreingestellte Größe des Systemgrößenveränderungsfeldes.

SBS_TOPALGIN

Wird mit dem Stil *SBS_HORZ* benutzt. Der obere Rand der Bildlaufleiste wird am oberen Rand des Rechtecks ausgerichtet, das durch die in der Funktion *CreateWindow* angegebenen Parameter *X*, *Y*, *Width* und *Height* festgelegt wird. Die Bildlaufleiste hat die voreingestellte Höhe für Systembildlaufleisten.

SBS_VERT

Bestimmt eine vertikale Bildlaufleiste. Ist weder der Stil *SBS_RIGHTALIGN* noch der Stil *SBS_-LEFTALIGN* angegeben, dann hat die Bildlaufleiste die in der Funktion *CreateWindow* angegebene Höhe, Breite und Position.

SC_XXXX

Die folgenden Befehle werden in der *WM_SYSCOMMAND*-Botschaft als Reaktion auf eine Auswahl im Systemmenü oder auf einen Klick im Verkleinerungs- bzw. Vergrößerungsfeld übergeben. Sie zeigen die Anforderung an.

SC_CLOSE

Fenster schließen.

SC_HSCROLL

Horizontalen Bildlauf durchführen.

SC_KEYMENU

Das dem Tastendruck entsprechende Menü darstellen.

SC_MAXIMIZE, SC_ZOOM	Fenster als Vollbild darstellen.
SC_MINIMIZE, SC_ICON	Fenster als Symbol darstellen.
SC_MOUSEMENU	Das dem Mausklick entsprechende Menü darstellen.
SC_MOVE	Das Fenster verschieben.
SC_NEXTWINDOW	In das nächste Fenster bewegen.
SC_PREVWINDOW	In das vorhergehende Fenster bewegen.
SC_RESTORE	Stellt nach einer Maximierung oder Minimierung die ursprüngliche Größe des Fensters wieder her.
SC_SIZE	Größe des Fensters einstellen.
SC_TASKLIST	Zeigt eine Task-Liste an.
SC_VSCROLL	Vertikalen Bildlauf durchführen.

Schnittstellenkonfigurations-Konstanten

Eine der folgenden Konstanten kann im Feld *Parity* einer *DCB*-Struktur verwendet werden. Sie zeigen den Paritätstyp an, der von einer seriellen Schnittstelle verwendet werden soll.

Konstante	Parität
EVENPARITY	Gerade
MARKPARITY	Markierung
NOPARITY	Keine
ODDPARITY	Ungerade
SPACEPARITY	Leer

Eine der folgenden Konstanten kann im Feld *StopBits* einer *DCB*-Struktur verwendet werden. Sie zeigt die Anzahl von Stopbits an, die von einer seriellen Schnittstelle verwendet werden.

Konstante	Verwendete Stopbits
ONESTOPBIT	1
ONE5STOPBITS	1,5
TWOSTOPBITS	2

Schriftdarstellungs-qualitäts-Flags

Diese Konstanten bestimmen die gewünschte Ausgabequalität für Schriften, die mit der Funktion *CreateFont* erzeugt werden.

DEFAULT_QUALITY	Ausgabequalität ist unerheblich.
DRAFT_QUALITY	Ausgabequalität ist von eingeschränktem Interesse.
PROOF_QUALITY	Die Ausgabequalität der Zeichen ist wichtiger als die Übereinstimmung mit virtuellen Schriftattributen.

ShowWindow-Befehle (frühere Versionen)

Dies sind ältere Konstanten, die nur aus Gründen der Abwärtskompatibilität mit früheren Windows-Versionen übernommen wurden.

Wenn möglich, sollten die *SW_*-Fensterdarstellungskonstanten verwendet werden.

Konstante	Wert	Bedeutung
HIDE_WINDOW	0	Fenster verbergen.
SHOW_OPENWINDOW	1	Ein Fenster aus seinem Symbol wiederherstellen.
SHOW_ICONWINDOW	2	Fenster zum Symbol verkleinern.
SHOW_FULLSCREEN	3	Fenster zum Vollbild vergrößern.
SHOW_OPENNOACTIVATE	4	Fenster wiederherstellen, aber nicht aktivieren.

SIZE_XXXX

Diese Konstanten bestimmen die Art der angeforderten Fenstergrößenänderung. Sie werden in *WM_SIZE*-Botschaften übergeben.

SIZEFULLSCREEN	Fenster wurde zum Vollbild vergrößert.
SIZEICONIC	Fenster wurde zum Symbol verkleinert.
SIZENORMAL	Die Größe des Fensters wurde geändert, *SIZEICONIC* oder *SIZEFULLSCREEN* treffen jedoch nicht zu.
SIZEZOOMHIDE	Die Botschaft wird an alle Pop-up-Fenster übergeben, wenn ein anderes Fenster zum Vollbild vergrößert wurde.

SIZEZOOMSHOW	Die Botschaft wird an alle Pop-up-Fenster übergeben, wenn ein anderes Fenster in seiner voreingestellten Größe wiederhergestellt wurde.

SM_XXXX

Diese Codes bestimmen Eigenschaften der Windows-Anwenderschnittstelle. Die jeweiligen Abmessungen können über die Funktion *GetSystemMetrics* geliefert werden.

SM_CXBORDER	Breite des Fensterrahmens, nicht einstellbar.
SM_CXCURSOR	Breite des Zeigers.
SM_CXDLGFRAME	Breite des Rahmens, wenn das Fenster den Stil *WS_DLGFRAME* hat.
SM_CXFRAME	Breite des Fensterrahmens, einstellbar.
SM_CXFULLSCREEN	Breite des Fenster-Client-Bereiches für eine Vollbildanwendung.
SM_CXHSCROLL	Breite des Pfeilbitmaps auf der horizontalen Bildlaufleiste.
SM_CXHTHUMB	Breite des Positionszeigers auf der horizontalen Bildlaufleiste.
SM_CXICON	Breite eines Symbols.
SM_CXMIN	Minimale Breite des Fensters.
SM_CXMINTRACK	Minimale Tracking-Breite des Fensters.
SM_CXSCREEN	Breite des Bildschirms.
SM_CXSIZE	Breite des in der Titelzeile enthaltenen Bitmap.
SM_CXVSCROLL	Breite des Pfeilbitmap auf der vertikalen Bildlaufleiste.
SM_CYBORDER	Höhe des Fensterrahmens, nicht einstellbar.
SM_CYCAPTION	Höhe der Titelzeile eines Fensters.
SM_CYCURSOR	Höhe des Zeigers.
SM_CYDLGFRAME	Höhe des Rahmens, wenn das Fenster den Stil *WS_DLGFRAME* hat.
SM_CYFRAME	Höhe des Fensterrahmens, einstellbar.
SM_CYFULLSCREEN	Höhe des Fenster-Client-Bereiches für eine Vollbildanwendung (gleich der Höhe des Bildschirms minus der Höhe der Titelzeile).

SM_CYHSCROLL	Höhe des Pfeilbitmaps auf der horizontalen Bildlaufleiste.
SM_CYICON	Höhe des Symbols.
SM_CYKANJIWINDOW	Höhe des Kanji-Fensters.
SM_CYMENU	Höhe einer einzeiligen Menüleiste.
SM_CYMIN	Minimale Höhe des Fensters.
SM_CYMINTRACK	Minimale Tracking-Höhe des Fensters.
SM_CYSCREEN	Höhe des Bildschirms.
SM_CYSIZE	Höhe des in der Titelzeile enthaltenen Bitmap.
SM_CYVSCROLL	Höhe des Pfeilbitmap auf der vertikalen Bildlaufleiste.
SM_CYVTHUMB	Höhe des Positionszeigers auf der vertikalen Bildlaufleiste.
SM_DEBUG	Ungleich Null, wenn die Windows-Debugger-Version vorliegt.
SM_MOUSEPRESENT	Ungleich Null, wenn eine Maus installiert ist.
SM_SWAPBUTTON	Ungleich Null, wenn die linke und rechte Maustaste vertauscht wurden.

SP_XXXX

Diese Konstanten werden als Fehlercodes durch die *Escape*-Funktion zurückgegeben.

SP_APPABORT	Die Anwendung hat den Druckauftrag abgebrochen.
SP_ERROR	Allgemeiner Fehler.
SP_OUTOFDISK	Speicherplatz auf der Diskette bzw. Platte nicht ausreichend für Spooling. Zusätzlicher Speicherplatz kann nicht bereitgestellt werden.
SP_OUTOFMEMORY	Arbeitsspeicher ist für Spooling nicht ausreichend.
SP_USERABORT	Der Benutzer hat den Druckauftrag mit Hilfe des Druck-Managers abgebrochen.

SS_XXXX

Diese Konstanten werden verwendet, um den Stil eines statischen Steuerelements festzulegen, wenn es durch die Funktionen *CreateWindow* oder *CreateWindowEx* erzeugt wird.

SS_BLACKFRAME	Bestimmt ein Feld mit einem Rahmen, der in der gleichen Farbe wie Fensterrahmen gezeichnet wird. Diese Farbe ist Schwarz im voreingestellten Windows-Farbschema.
SS_BLACKRECT	Bestimmt ein Rechteck, das mit der gleichen Farbe gefüllt ist, mit der Fensterrahmen gezeichnet werden. Diese Farbe ist Schwarz im voreingestellten Windows-Farbschema.
SS_CENTER	Bestimmt ein einfaches Rechteck und zeigt den gegebenen Text zentriert im Rechteck an. Der Text wird formatiert, bevor er angezeigt wird. Wörter, die über das Ende einer Zeile hinausreichen würden, werden automatisch auf den Anfang der nächsten zentrierten Zeile umbrochen.
SS_GRAYFRAME	Bestimmt ein Feld mit einem Rahmen, der in der gleichen Farbe wie der Bildschirmhintergrund (Desktop) gezeichnet wird.
SS_GRAYRECT	Bestimmt ein Rechteck, das mit der Farbe gefüllt wird, die verwendet wird, um den Bildschirmhintergrund zu zeichnen. Dies ist Grau im voreingestellten Windows-Farbschema.
SS_ICON	Bestimmt ein Symbol, das im Dialogfenster angezeigt wird. Der gegebene Text ist der Name eines Symbols (kein Dateiname), der anderswo in der Ressourcendatei definiert ist. Die Parameter *Width* und *Height* werden ignoriert; das Symbol legt seine Größe selbst fest.
SS_LEFT	Bestimmt ein einfaches Rechteck und zeigt den gegebenen Text linksbündig im Rechteck an. Der Text wird formatiert, bevor er angezeigt wird. Wörter, die über das Ende einer Zeile hinausreichen würden, werden automatisch auf den Anfang der nächsten Zeile umbrochen.
SS_LEFTNOWORDWRAP	Bestimmt ein einfaches Rechteck und zeigt den gegebenen Text linksbündig im Rechteck an. Tabulatorzeichen werden erweitert, aber Wörter werden nicht umbrochen. Text, der über das Ende einer Zeile hinausreicht, wird abgeschnitten.

SS_NOPREFIX

Wenn dieser Stil nicht angegeben wird, wird Windows jedes »&« Zeichen im Text des Steuerelements als Akzelerator-Präfixzeichen interpretieren. In diesem Fall wird das »&« entfernt und das nächste Zeichen im String unterstrichen. Soll ein statisches Steuerelement Text enthalten, bei dem dieses Merkmal unerwünscht ist, kann *SS_NOPREFIX* hinzugefügt werden.

Dieser statische Steuerelementstil kann in jedes der definierten Steuerelemente einbezogen werden. Man kann *SS_NOPREFIX* mit anderen Stilen kombinieren, indem der bitweise ODER-Operator (|) verwendet wird. Dies wird meist genutzt, wenn Dateinamen oder andere Strings, die ein »&« enthalten können, in einem statischen Steuerelement in einem Dialogfenster angezeigt werden sollen.

SS_RIGHT

Bestimmt ein einfaches Rechteck und zeigt den gegebenen Text rechtsbündig im Rechteck an. Der Text wird formatiert, bevor er angezeigt wird. Wörter, die über das Ende einer Zeile hinausreichen würden, werden automatisch auf die nächste rechtsbündige Zeile umgebrochen.

SS_SIMPLE

Bestimmt ein einfaches Rechteck und zeigt eine einzelne Textzeile linksbündig im Rechteck an. Die Textzeile kann auf keinen Fall gekürzt oder verändert werden. Das übergeordnete Fenster oder das Dialogfenster des Steuerelements darf die Botschaft *WM_CTLCOLOR* nicht bearbeiten.

SS_USERITEM

Bestimmt ein anwenderdefiniertes, statisches Steuerelement.

SS_WHITEFRAME

Bestimmt ein Feld mit einem Rahmen, der in der gleichen Farbe wie der Fensterhintergrund gezeichnet wird. Diese Farbe ist Weiß im voreingestellten Windows-Farbschema.

SS_WHITERECT

Bestimmt ein Rechteck, das mit der Farbe gefüllt wird, die auch zur Darstellung des Fensterhintergrundes verwendet wird. Diese Farbe ist Weiß im voreingestellten Windows-Farbschema.

StretchBlt-Modi

Diese Konstanten sind die Bitmap-Streckmodi. Sie werden in den Funktionen *GetStretchBltMode* und *SetStretchBltMode* verwendet.

BLACKONWHITE	AND (&) in den getilgten Zeilen. Dieser Modus erhält schwarze auf Kosten der weißen Pixel durch Verwendung des UND-Operators (&) mit den zu tilgenden und den verbleibenden Zeilen.
COLORONCOLOR	Löscht die getilgten Zeilen. Dieser Modus löscht alle getilgten Zeilen und versucht dabei nicht, ihre Information zu erhalten.
WHITEONBLACK	ODER (|) in den getilgten Zeilen. Dieser Modus erhält weiße Pixel auf Kosten der schwarzen durch Verwendung des ODER-Operators (|) mit den zu tilgenden und den verbleibenden Zeilen.

SW_XXXX

Diese Konstanten bestimmen den Status, mit dem die Funktion *ShowWindow* Fenster darstellt.

SW_HIDE	Versteckt das Fenster und gibt die Aktivierung an ein anderes Fenster ab.
SW_MAXIMIZE	Wie *SW_SHOWMAXIMIZED*.
SW_MINIMIZE	Verkleinert das festgelegte Fenster und aktiviert das Hauptfenster in der Liste der Fenster-Verwaltung.
SW_NORMAL	Wie *SW_SHOWNORMAL*.
SW_OTHERZOOM	Ein anderes Fenster wird vergrößert. Wegen der Kompatibilität mit Windows 2.0 übernommen.
SW_OTHERUNZOOM	Ein anderes Fenster wird verkleinert. Wegen der Kompatibilität mit Windows 2.0 übernommen.
SW_RESTORE	Wie *SW_SHOWNORMAL*.
SW_SHOW	Aktiviert ein Fenster und zeigt es in seiner aktuellen Größe und Position.
SW_SHOWMAXIMIZED	Aktiviert das Fenster und zeigt es als vergrößertes Fenster.
SW_SHOWMINIMIZED	Aktiviert ein Fenster und zeigt es als Symbol.

SW_SHOWMINNOACTIVE Zeigt das Fenster als Symbol. Das gegenwärtig aktive Fenster bleibt aktiv.

SW_SHOWNA Zeigt das Fenster in seinem aktuellen Zustand. Das gegenwärtig aktive Fenster bleibt aktiv.

SW_SHOWNOACTIVATE Zeigt ein Fenster in seiner neuesten Größe und Position. Das gegenwärtig aktive Fenster bleibt aktiv.

SW_SHOWNORMAL Aktiviert ein Fenster und zeigt es an. Ist das Fenster verkleinert oder vergrößert, so stellt Windows es in seiner ursprünglichen Größe und Position wieder her.

Diese Bezeichner bestimmen den Status des darzustellenden Fensters und werden in der *WM_SHOWWINDOW*-Botschaft übergeben.

SW_PARENTCLOSING Es wird ein übergeordnetes Fenster verkleinert (Symbol) oder ein Pop-up-Fenster verborgen.

SW_PARENTOPENING Es wird ein übergeordnetes Fenster geöffnet oder ein Pop-up-Fenster gezeigt.

SWP_XXXX

Diese Flags werden in der Funktion *SetWindowPos* bzw. *DeferWindowPos* übergeben, um besondere Aktionen für das angegebene Fenster zu vereinbaren.

SWP_DRAWFRAME Zeichnet den in der Fensterklassendefinition definierten Rahmen um das Fenster.

SWP_HIDEWINDOW Versteckt das Fenster.

SWP_NOACTIVATE Betrifft nicht das aktivierte Fenster.

SWP_NOMOVE Erhält die aktuelle Position, ignoriert die Parameter *x* und *y*.

SWP_NOREDRAW Änderungen werden nicht dargestellt.

SWP_NOSIZE Erhält die aktuelle Größe, ignoriert die Parameter *cx* und *cy*.

SWP_NOZORDER Erhält die aktuelle Ordnung, ignoriert den Parameter *WndInsertAfter*.

SWP_SHOWWINDOW Stellt das Fenster dar.

SYSPAL_XXXX

Diese Konstanten bestimmen die neue Verwendung der Systempalette. Sie werden in den Funktionen *GetSystemPaletteUse* und *SetSystemPaletteUse* übergeben.

SYSPAL_NOSTATIC	Die Systempalette enthält keine statischen Farben außer Schwarz und Weiß.
SYSPAL_STATIC	Die Systempalette enthält statische Farben, die sich nicht ändern, wenn eine Anwendung ihre virtuelle Palette realisiert.

TA_XXXX

Diese Ausrichtungsoptionen steuern die Ausrichtung von Text, der mit einer der Funktionen *TextOut* oder *ExtTextOut* dargestellt wird. Die Optionen werden in einer der Funktionen *GetTextAlign* oder *SetTextAlign* festgelegt oder zurückgegeben.

TA_BASELINE	Bestimmt die Ausrichtung der x-Achse an der Grundlinie der ausgewählten Schrift im umgebenden Rechteck.
TA_BOTTOM	Bestimmt die Ausrichtung der x-Achse an der Unterseite des umgebenden Rechtecks.
TA_CENTER	Bestimmt die Ausrichtung der y-Achse am Mittelpunkt des umgebenden Rechtecks.
TA_LEFT	Bestimmt die Ausrichtung der y-Achse an der linken Seite des umgebenden Rechtecks.
TA_NOUPDATECP	Bestimmt, daß die aktuelle Position nicht nach jedem Aufruf der Funktionen *TextOut* oder *ExtTextOut* aktualisiert wird.
TA_RIGHT	Bestimmt die Ausrichtung der y-Achse an der rechten Seite des umgebenden Rechtecks.
TA_TOP	Bestimmt die Ausrichtung der x-Achse an der Oberseite des umgebenden Rechtecks.
TA_UPDATECP	Bestimmt, daß die aktuelle Position nach jedem Aufruf der Funktionen *TextOut* oder *ExtTextOut* aktualisiert wird.

TC_XXXX

Diese Konstanten repräsentieren die Textdarstellungsfähigkeiten eines Geräts.

TC_CP_STROKE	Gerät kann vektoriell mit Clipping ausgeben.
TC_CR_90	Gerät kann Buchstaben um 90 Grad rotieren.
TC_CR_ANY	Gerät kann beliebige Buchstabenrotationen vornehmen.
TC_EA_DOUBLE	Gerät kann Zeichen fett darstellen.
TC_IA_ABLE	Gerät kann kursive Zeichen darstellen.
TC_OP_CHARACTER	Gerät kann Zeichen genau ausgeben.
TC_OP_STROKE	Gerät kann Zeichen vektoriell ausgeben.
TC_RA_ABLE	Gerät kann Rasterschriften darstellen.
TC_SA_CONTIN	Gerät kann einen beliebigen vielfachen Wert zur Skalierung verwenden.
TC_SA_DOUBLE	Gerät kann mit zweifachem Zeichenabstand skalieren.
TC_SA_INTEGER	Gerät kann das Mehrfache eines Integerwerts zur Skalierung verwenden.
TC_SF_X_YINDEP	Gerät kann unabhängig von X und Y skalieren.
TC_SO_ABLE	Gerät kann Zeichen durchgestrichen darstellen.
TC_UA_ABLE	Gerät kann unterstreichen.
TC_VA_ABLE	Gerät kann Vektorschriften darstellen.

Ternäre Rasteroperationen

Die folgenden Konstanten werden als Rasteroperationscodes in den Funktionen *BitBlt*, *PatBlt*, *StretchBlt* und *StretchDIBits* verwendet.

BLACKNESS	Alle Ausgaben werden schwarz dargestellt.
DSTINVERT	Invertiert das Ziel-Bitmap.
MERGECOPY	Kombiniert Vorgabe- und Quell-Bitmap unter Verwendung des logischen UND-Operators (&&).
MERGEPAINT	Kombiniert das invertierte Quell-Bitmap mit dem Ziel-Bitmap unter Verwendung des logischen ODER-Operators (||).

NOTSRCCOPY	Kopiert das invertierte Ziel-Bitmap ins Ziel.
NOTSRCERASE	Invertiert das Resultat der Kombination von Quell- und Ziel-Bitmap unter Verwendung des logischen ODER-Operators (||), also das Resultat von MERGEPAINT.
PATCOPY	Kopiert die Vorgabe in das Ziel-Bitmap.
PATINVERT	Kombiniert das Ziel-Bitmap mit der Vorgabe unter Verwendung des logischen Exclusives-ODER-Operators.
PATPAINT	Kombiniert das invertierte Quell-Bitmap mit der Vorgabe unter Verwendung des logischen ODER-Operators (||). Das Resultat wird mit dem Ziel-Bitmap unter Verwendung des logischen ODER-Operators (||) kombiniert.
SRCAND	Kombiniert Pixel des Ziel- und Quell-Bitmap unter Verwendung des logischen UND-Operators (&&).
SRCCOPY	Kopiert das Quell- in das Ziel-Bitmap.
SRCERASE	Invertiert das Ziel-Bitmap und kombiniert das Resultat mit dem Quell-Bitmap unter Verwendung des logischen UND-Operators (&&).
SRCINVERT	Kombiniert Pixel von Ziel- und Quell-Bitmap unter Verwendung des logischen Exclusives-ODER-Operators.
SRCPAINT	Kombiniert Pixel von Ziel- und Quell-Bitmap unter Verwendung des logischen ODER-Operators (||).
WHITENESS	Färbt jede Ausgabe weiß.

TF_FORCEDRIVE

Wird dieses Flag mit dem Argument *DriveLetter* in einem Aufruf der Funktion *GetTempFileName* kombiniert, stellt es sicher, daß die temporäre Datei auf dem angegebenen Laufwerk erzeugt wird. Andernfalls wird diese Datei auf der ersten Festplatte oder im Verzeichnis der Environment-Variablen TEMP erzeugt.

VK_XXXX

Virtuelle Tastencodes sind Konstanten, die zur Repräsentierung der Standardtasten (z.B. »A« oder »F1«) des Computers verwendet werden. Da verschie-

dene Computermodelle unterschiedliche Tasten haben können, werden die virtuellen Tastencodes verwendet, um die Lauffähigkeit einer Anwendung auf verschiedenen Computern zu ermöglichen. Diese Codes werden zur Definition von Akzeleratoren oder zur Bearbeitung der Tastaturbotschaften *WM_KEYDOWN*, *WM_KEYUP*, *WM_SYSKEYDOWN* und *WM_SYSKEYUP* verwendet.

Die folgende Tabelle zeigt die virtuellen Tastencodes und die ihnen entsprechenden Tasten.

Bezeichnung	Wert	Beschreibung
VK_LBUTTON	0x01	Linke Maustaste
VK_RBUTTON	0x02	Rechte Maustaste
VK_CANCEL	0x03	Für die Verarbeitung von Control-Break
VK_MBUTTON	0x04	Mittlere Maustaste (Dreitastenmaus)
	0x05-0x07	Nicht definiert
VK_BACK	0x08	Backspace-Taste
VK_TAB	0x09	Tab-Taste
	0x0A-0x0B	Nicht definiert
VK_CLEAR	0x0C	Löschtaste (Bildschirm)
VK_RETURN	0x0D	Return-Taste
	0x0E-0x0F	Nicht definiert
VK_SHIFT	0x10	Shift-Taste
VK_CONTROL	0x11	Control-Taste
VK_MENU	0x12	Menütaste
VK_PAUSE	0x13	Pause-Taste
VK_CAPITAL	0x14	CapsLock-Taste
	0x15-0x19	Reserviert für Kanji-Systeme
	0x1A	Nicht definiert
VK_ESCAPE	0x1B	Escape-Taste
	0xC-0x1F	Rerserviert für Kanji-Systeme
VK_SPACE	0x20	Leertaste
VK_PRIOR	0x21	PgUp-Taste
VK_NEXT	0x22	PgDn-Taste
VK_END	0x23	End-Taste
VK_HOME	0x24	Home-Taste
VK_LEFT	0x25	Cursortaste Links
VK_UP	0x26	Cursortaste Oben
VK_RIGHT	0x27	Cursortaste Rechts
VK_DOWN	0x28	Cursortaste Unten
VK_SELECT	0x29	Auswahltaste (Select)
VK_PRINT	0x2A	OEM-spezifisch
VK_EXECUTE	0x2B	Taste Ausführen (Execute)
VK_SNAPSHOT	0x2C	PrintScreen für Windows Version 3.0 und höher
VK_INSERT	0x2D	Ins-Taste
VK_DELETE	0x2E	Del-Taste
VK_HELP	0x2F	Hilfetaste
VK_0	0x30	Taste 0

Bezeichnung	Wert	Beschreibung
VK_1	0x31	Taste 1
VK_2	0x32	Taste 2
VK_3	0x33	Taste 3
VK_4	0x34	Taste 4
VK_5	0x35	Taste 5
VK_6	0x36	Taste 6
VK_7	0x37	Taste 7
VK_8	0x38	Taste 8
VK_9	0x39	Taste 9
	0x3A-0x40	Nicht definiert
VK_A	0x41	Taste A
VK_B	0x42	Taste B
VK_C	0x43	Taste C
VK_D	0x44	Taste D
VK_E	0x45	Taste E
VK_F	0x46	Taste F
VK_G	0x47	Taste G
VK_H	0x48	Taste H
VK_I	0x49	Taste I
VK_J	0x4A	Taste J
VK_K	0x4B	Taste K
VK_L	0x4C	Taste L
VK_M	0x4D	Taste M
VK_N	0x4E	Taste N
VK_O	0x4F	Taste O
VK_P	0x50	Taste P
VK_Q	0x51	Taste Q
VK_R	0x52	Taste R
VK_S	0x53	Taste S
VK_T	0x54	Taste T
VK_U	0x55	Taste U
VK_V	0x56	Taste V
VK_W	0x57	Taste W
VK_X	0x58	Taste X
VK_Y	0x59	Taste Y
VK_Z	0x5A	Taste Z
	0x5B-0x5F	Nicht definiert
VK_NUMPAD0	0x60	Numerisches Tastenfeld, Taste 0
VK_NUMPAD1	0x61	Numerisches Tastenfeld, Taste 1
VK_NUMPAD2	0x62	Numerisches Tastenfeld, Taste 2
VK_NUMPAD3	0x63	Numerisches Tastenfeld, Taste 3
VK_NUMPAD4	0x64	Numerisches Tastenfeld, Taste 4
VK_NUMPAD5	0x65	Numerisches Tastenfeld, Taste 5
VK_NUMPAD6	0x66	Numerisches Tastenfeld, Taste 6
VK_NUMPAD7	0x67	Numerisches Tastenfeld, Taste 7
VK_NUMPAD8	0x68	Numerisches Tastenfeld, Taste 8

Bezeichnung	Wert	Beschreibung
K_NUMPAD9	0x69	Numerisches Tastenfeld, Taste 9
VK_MULTIPLY	0x6A	Multiplikationstaste
VK_ADD	0x6B	Additionstaste
VK_SEPARATER	0x6C	Trenntaste
VK_SUBTRACT	0x6D	Subtraktionstaste
VK_DECIMAL	0x6E	Dezimaltrennzeichentaste
VK_DIVIDE	0x6F	Divisionstaste
VK_F1	0x70	F1
VK_F2	0x71	F2
VK_F3	0x72	F3
VK_F4	0x73	F4
VK_F5	0x74	F5
VK_F6	0x75	F6
VK_F7	0x76	F7
VK_F8	0x77	F8
VK_F9	0x78	F9
VK_F10	0x79	F10
VK_F11	0x7A	F11
VK_F12	0x7B	F12
VK_F13	0x7C	F13
VK_F14	0x7D	F14
VK_F15	0x7E	F15
VK_F16	0x7F	F16
	0x80-0x87	OEM-spezifisch
	0x88-0x8F	Nicht definiert
VK_NUMLOCK	0x90	Taste NumLock
VK_OEM_SCROLL	0x91	Taste ScrollLock
	0x92-0xB9	Nicht definiert
VK_OEM_1	0xBA	Tastaturspezifische Interpunktionstaste (nicht auf jeder Tastatur vorhanden)
VK_OEM_PLUS	0xBB	Taste Plus +
VK_OEM_COMMA	0xBC	Taste Komma ,
VK_OEM_MINUS	0xBD	Taste Minus -
VK_OEM_PERIOD	0xBE	Taste Punkt .
VK_OEM_2	0xBF	Tastaturspezifische Interpunktionstaste (nicht auf jeder Tastatur vorhanden)
VK_OEM_3	0xC0	Tastaturspezifische Interpunktionstaste (nicht auf jeder Tastatur vorhanden)
	0xC1-0xDA	Nicht definiert
VK_OEM_4	0xDB	Tastaturspezifische Interpunktionstaste (nicht auf jeder Tastatur vorhanden)
VK_OEM_5	0xDC	Tastaturspezifische Interpunktionstaste (nicht auf jeder Tastatur vorhanden)
VK_OEM_6	0xDD	Tastaturspezifische Interpunktionstaste (nicht auf jeder Tastatur vorhanden)

Bezeichnung	Wert	Beschreibung
VK_OEM_7	0xDE	Tastaturspezifische Interpunktionstaste (nicht auf jeder Tastatur vorhanden)
VK_OEM_8	0xDF	Tastaturspezifische Interpunktionstaste (nicht auf jeder Tastatur vorhanden)
	0xE0-0xE1	OEM-spezifisch
VK_OEM_102	0xE2	<> oder | auf erweiterter, nicht US-IBM-kompatibler Tastatur mit 102-Tasten
	0xE3-0xE4	OEM-spezifisch
	0xE5	Nicht definiert
	0xE6	OEM-spezifisch
	0xE7-0xE8	Nicht definiert
	0xE9-0xF5	OEM-spezifisch
	0xF6-0xFE	Nicht definiert

Vordefinierte Zeichenobjekte

Diese Konstanten repräsentieren vordefinierte (Stock) Zeichenobjekte. Sie werden in der Funktion *GetStockObject* verwendet.

BLACK_BRUSH	Schwarzer Pinsel.
DKGRAY_BRUSH	Dunkelgrauer Pinsel.
GRAY_BRUSH	Grauer Pinsel.
HOLLOW_BRUSH	Neutraler Pinsel (zeichnet nicht).
LTGRAY_BRUSH	Hellgrauer Pinsel.
NULL_BRUSH	Neutraler Pinsel.
WHITE_BRUSH	Weißer Pinsel.
BLACK_PEN	Schwarzer Stift.
NULL_PEN	Neutraler Stift (zeichnet nicht).
WHITE_PEN	Weißer Stift.
ANSI_FIXED_FONT	ANSI-Systemschrift mit festen Zeichenabständen.
ANSI_VAR_FONT	ANSI-Systemschrift mit variablen Zeichenabständen.
DEVIDE_DEFAULT_FONT	Geräteabhängige Schrift.
OEM_FIXED_FONT	OEM-abhängige Schrift mit festen Zeichenabständen.
SYSTEM_FIXED_FONT	Die Systemschrift mit festem Zeichenabstand, die in früheren Windows-Versionen benutzt

wurde. Nur aus Gründen der Kompatibilität verfügbar.

SYSTEM_FONT Die Systemschrift. Standardmäßig verwendet Windows die Systemschrift, um Menüs, Dialogfenster und anderen Text zu zeichnen.
In den Windows-Versionen 3.0 und höher ist die Systemschrift eine Proportionalschrift.

DEFAULT_PALETTE Standardfarbpalette. Diese Farbpalette besteht aus den 20 statischen Farben, die immer in der Systempalette präsent sind, um Farben in der virtuellen Palette von Hintergrundfenstern anzugleichen.

WEP_XXXX

Diese Codes werden in der Variablen *ExitCode* verwendet, auf die aus einer dynamischen Linkbibliothek zugegriffen werden kann.

WEP_FREE_DLL Zeigt an, daß nur die dynamische Linkbibliothek beendet und aus dem Speicher entfernt wird.

WEP_SYSTEM_EXIT Zeigt an, daß Windows beendet wird.

WF_XXXX

Diese Konstanten bestimmen die aktuelle Speicherkonfiguration von Windows. Sie werden als Rückgabewerte der Funktion *GetWinFlags* verwendet.

WF_80X87 System besitzt einen mathematischen Coprozessor.

WF_CPU086 System-CPU ist ein 8086.

WF_CPU186 System-CPU ist ein 80186.

WF_CPU286 System-CPU ist ein 80286.

WF_CPU386 System-CPU ist ein 80386.

WF_CPU486 System-CPU ist ein 80486.

WF_ENHANCED Windows läuft im erweiterten 386-Modus. Das Flag *WF_PMODE* ist immer gesetzt, wenn *WF_ENHANCED* gesetzt ist.

WF_LARGEFRAME Windows läuft mit Large-Frame-EMS-Speicherkonfiguration.

WF_PMODE	Windows läuft im Protected Mode. Dieses Flag ist immer dann gesetzt, wenn entweder *WF_ENHANCED* oder *WF_STANDARD* gesetzt ist.
WF_SMALLFRAME	Windows läuft mit Small-Frame-EMS-Speicherkonfiguration.
WF_STANDARD	Windows läuft im Standard Modus. *WF_PMODE* ist immer gesetzt, wenn *WF_STANDARD* gesetzt ist.
WF_WIN286	Wie *WF_STANDARD*.
WF_WIN386	Wie *WF_ENHANCED*.

WH_XXXX

Diese Codes legen einen bestimmten Typ der zu installierenden Filterfunktion fest (*SetWindowsHook*) oder entfernen ihn (*UnhookWindowsHook*) aus der Kette der Filterfunktionen.

WH_CALLWNDPROC	Fensterfunktionsfilter.
WH_GETMESSAGE	Botschaftsfilter.
WH_JOURNALPLAYBACK	Wiedergabeprotokollfilter.
WH_JOURNALRECORD	Aufzeichnungsprotokollfilter.
WH_KEYBOARD	Tastaturfilter.
WH_MSGFILTER	Botschaftsfilter (nur *SetWindowsHook*).
WH_SYSMSGFILTER	Systemweiter Botschaftsfilter (nur *SetWindowsHook*).

WS_XXXX

Diese Konstanten werden in verschiedenen Kombinationen verwendet, um Fensterstile festzulegen, wenn Fenster, Dialogfenster oder Steuerelemente mit der Funktion *CreateWindow* oder *CreateWindowEx* erzeugt werden.

WS_BORDER	Erzeugt ein Fenster, das eine Umrandung hat. Dieser Stil kann nicht zusammen mit *WS_DLGFRAME* verwendet werden.
WS_CAPTION	Erzeugt ein Fenster, das eine Titelleiste hat (hat den Stil *WS_BORDER* zur Folge). Dieser Stil kann nicht mit dem Stil *WS_DLGFRAME* verwendet werden.

WS_CHILD Erzeugt ein untergeordnetes Fenster. Kann nicht mit dem Stil *WS_POPUP* verwendet werden.

WS_CHILDWINDOW Erzeugt ein untergeordnetes Fenster, das den Stil *WS_CHILD* hat.

WS_CLIPCHILDREN Schließt den von untergeordneten Fenstern belegten Bereich aus, wenn innerhalb des übergeordneten Fensters gezeichnet wird. Wird verwendet, wenn das übergeordnete Fenster erzeugt wird.

WS_CLIPSIBLINGS Schneidet untergeordnete Fenster relativ zueinander ab; das heißt, wenn ein bestimmtes untergeordnetes Fenster eine *WM_PAINT*-Botschaft erhält, schneidet der Stil *WS_CLIPSIBLINGS* alle anderen überlappten untergeordneten Fenster aus der Region des untergeordneten Fensters heraus, das aktualisiert werden soll.

Wenn *WS_CLIPSIBLINGS* nicht gesetzt ist und untergeordnete Fenster überlappen sich, so ist es beim Zeichnen innerhalb des Client-Bereichs eines untergeordneten Fensters möglich, auch innerhalb des Client-Bereichs eines benachbarten untergeordneten Fensters zu zeichnen. Nur zum Gebrauch mit dem Stil *WS_CHILD*.

WS_DISABLED Erzeugt ein Fenster, das anfänglich nicht aktiviert ist.

WS_DLGFRAME Erzeugt ein Fenster mit einer doppelten Umrandung, aber ohne Titelleiste. Kann nicht mit *WS_BORDER* kombiniert werden.

WS_GROUP Bestimmt das erste Steuerelement aus einer Gruppe von Elementen, in der der Anwender sich mit den Richtungstasten von einem Steuerelement zum nächsten bewegen kann. Alle Steuerelemente, die nach dem ersten Element mit dem Stil *WS_GROUP* definiert wurden, gehören zur gleichen Gruppe. Das nächste Steuerelement mit dem Stil *WS_GROUP* beendet die Stilgruppe und beginnt die nächste Gruppe (das heißt, eine Gruppe hört da auf, wo die nächste beginnt). Nur Dialogfenster verwenden diesen Stil.

WS_HSCROLL Erzeugt ein Fenster, das eine horizontale Bildlaufleiste hat.

WS_ICONIC Erzeugt ein Fenster, das anfangs als Symbol dar-

	gestellt wird. Nur zum Gebrauch mit dem Stil *WS_OVERLAPPED*.
WS_MAXIMIZE	Erzeugt ein Fenster von maximaler Größe.
WS_MAXIMIZEBOX	Erzeugt ein Fenster, das ein Feld zur Ausdehnung auf größtmöglichen Umfang hat.
WS_MINIMIZE	Erzeugt ein Fenster, das ursprünglich als Symbol dargestellt wird. Kann nur zusammen mit dem Stil *WS_OVERLAPPED* verwendet werden.
WS_MINIMIZEBOX	Erzeugt ein Fenster, das ein Feld zur Reduzierung zum Symbol hat.
WS_OVERLAPPED	Erzeugt ein überlapptes Fenster. Ein überlapptes Fenster hat eine Überschrift und eine Umrandung.
WS_OVERLAPPEDWINDOW	Erzeugt ein überlapptes Fenster, das die Stile *WS_OVERLAPPED*, *WS_CAPTION*, *WS_SYSMENU*, *WS_THICKFRAME*, *WS_MINIMIZEBOX* und *WS_MAXIMIZEBOX* hat.
WS_POPUP	Erzeugt ein Pop-up-Fenster. Kann nicht mit dem Stil *WS_CHILD* benutzt werden.
WS_POPUPWINDOW	Erzeugt ein Pop-up-Fenster, das die Stile *WS_BORDER*, *WS_POPUP* und *WS_SYSMENU* hat. Der Stil *WS_CAPTION* muß mit dem Stil *WS_POPUPWINDOW* kombiniert werden, um das Systemmenü sichtbar zu machen.
WS_SIZEBOX	Wie *WS_THICKFRAME*.
WS_SYSMENU	Erzeugt ein Fenster, das ein Systemmenü in seiner Titelleiste hat.
WS_TABSTOP	Bestimmt eines der Steuerelemente, durch die sich der Anwender mit der Tab-Taste bewegen kann. Die Tab-Taste bewegt den Fokus zum nächsten Steuerelement, das den Stil *WS_TABSTOP* hat. Nur Dialogfenster verwenden den Stil *WS_TABSTOP*.
WS_THICKFRAME	Erzeugt ein Fenster mit einem breiten Rahmen, der verwendet werden kann, um die Größe des Fensters zu verändern.
WS_TILED	Wie *WS_OVERLAPPED*.
WS_TILEDWINDOW	Wie *WS_OVERLAPPEDWINDOW*.
WS_VISIBLE	Erzeugt ein Fenster, das anfänglich unsichtbar ist. Dies wird auf überlappte und Pop-up-Fenster angewendet.

WS_VSCROLL Erzeugt ein Fenster, das eine vertikale Bildlaufleiste hat.

WS_EX_XXXX

Diese Konstanten werden im Parameter *ExStyle* verwendet, um den erweiterten Stil eines Fensters festzulegen, wenn es mit der Funktion *CreateWindowEx* erzeugt wird.

WS_EX_DLGMODALFRAME Bestimmt ein Fenster mit einer doppelten Umrandung, das wahlweise mit einer Titelleiste erzeugt werden kann, indem man den Stil *WS_CAPTION* im Parameter *Style* angibt.

WS_EX_NOPARENTNOTIFY Bestimmt, daß ein untergeordnetes Fenster, das mit diesem Stil erzeugt wird, keine Botschaft *WM_PARENTNOTIFY* an sein übergeordnetes Fenster sendet, wenn das untergeordnete Fenster erzeugt oder zerstört wird.

Zeichenabstand-Flags

Diese Konstanten bestimmen den Zeichenabstand (proportional oder fest) für Schriften, die mit der Funktion *CreateFont* erzeugt werden.

DEFAULT_PITCH
FIXED_PITCH
VARIABLE_PITCH

Zeichensatz-Flags

Diese Konstanten bestimmen den gewünschten Zeichensatz für Schriften, die mit der Funktion *CreateFont* erzeugt werden.

ANSI_CHARSET *SHIFTJIS_CHARSET*
SYMBOL_CHARSET *OEM_CHARSET*

Index

A
Abbildungsmodus 592
  aktueller 181
  festlegen 344
Abmessungen
  Grafikfenster 217, 364
  Schrift 214
AccessResource-Funktion 44
AddAtom-Funktion 45
AddFontResource-Funktion 45
AdjustWindowRect-Funktion 46
AdjustWindowRectEx-Funktion 47
Aktionsschalter (vgl. Schalter)
  Abmessungen 479
  ausgewählt 253
aktivieren
  untergeordnete Fenster 449
Aktualisierungsregion
  Fenster 216
  Rechtecke 215
Akzeleratoren
  Befehle und 449
  laden 258
  übersetzen 395 f.
  virtuelle Tastencodes und 616 f., 619
AllocDStoCSAlias-Funktion 47
AllocResource-Funktion 48
AllocSelector-Funktion 48
alte ShowWindow-Konstanten 607
AnimatePalette-Funktion 49
AnsiLower-Funktion 50
AnsiLowerBuff-Funktion 50
AnsiNext-Funktion 50
AnsiPrev-Funktion 51
AnsiToOem-Funktion 51
AnsiToOemBuff-Funktion 51
AnsiUpper-Funktion 52
AnsiUpperBuff-Funktion 52
Anwendungen
  abfragen 495
  ausführen 263, 406 f.
  beenden 460, 496
  Fenster aktivieren 445
  Tasten überladen 462
AnyPopup-Funktion 53
AppendMenu-Funktion 53
Arc-Funktion 54
ArrangeIconicWindows-Funktion 55
ATOM-Typ 514
Atome 514
  auffinden 141, 227
  globale 223, 226 f., 230
  Handle 149
  hinzufügen 45
  löschen 105, 226
  Name 230
  Tabelle 242
Ausführung
  übergeben 415
Ausführungsumgebung
  wiederherstellen 392
Ausrichtung
  Text 362
Auswahlfelder
  Auswahlmarkierung 61 f., 181
  markieren 418
  Status 417 f.
Auswahlmarkierungen
  Größe 181
  Menüoptionen und 62
  Schalter und 61
  Schaltfelder 62

B
Basiseinheiten
  Dialogfenster 169
Befehle
  Akzelerator-Tasten und 449
  Bildlaufleisten 603
  Hilfe 579

Menüs 449
BeginDeferWindowPos-Funktion 55
BeginPaint-Funktion 56
Benachrichtigungscodes
Editierfelder 569
Kombinationsfenster 550
Listen 584
Schalter 547
Benachrichtigungsfensterflags 587
Bereiche
Bildlaufleisten 200, 354
Bewegungen
Funktionen 28
Bezeichner
Farbtabelle 559
BI_XXXX-Konstanten 547
Bibliotheken
freigeben 146
Funktionsadressen 195
laden 262
Bildlauf
Funktionen 29
Bildlaufleisten
aktuelle Position 200
ausblenden 382
Befehle 603
Bereiche 354
Bereiche abrufen 200
blättern 465
darstellen 382
klicken 510
Konstanten 603
Position 353
Stile 604
Bildlaufleistenbereiche
abrufen 200
Bildschirmkontexte
Bildlauf durchführen 320
Client-Bereiche abrufen 164
Fenster 218
Handle 532
BitBlt-Funktion 56f
BITMAP-Typ 514
BITMAPCOREHEADER-Typ 514
BITMAPCOREINFO-Typ 515
BITMAPFILEHEADER-Typ 516
BITMAPINFO-Typ 517
BITMAPINFOHEADER-Typ 517
Bitmaps 514, 516 f.
Bits 327, 338
Bits kopieren 150
Datei 516
erzeugen 69, 71
Farbtabelle kombinieren 515
gerätespezifische 79
geräteunabhängige 170
Größe 150, 327
Größe und Farben definieren 514
Handle 532
Informationen 517
kompatibel zum Gerätekontext 71
Kompressionskonstanten 547
komprimieren 385
kopieren 56f
laden 259
löschen 106
Menüoptionen 345
mit Attribut »discardable« erstellen 81
Pinsel erstellen aus 80
strecken 385, 612
verschieben 385
vordefinierte 593
BM_GETCHECK-Botschaft 417
BM_GETSTATE-Botschaft 417
BM_SETCHECK-Botschaft 418
BM_SETSTATE-Botschaft 418
BM_SETSTYLE-Botschaft 419
BN_XXXX-Konstanten 547
Bogenzeichnungsfähigkeiten 552
BOOL-Typ 518
Botschaften 540
abrufen 184
Anwendungen 305
Beendigung 307
Befehle 449
Dialogfenster und 252
dynamischer Datenaustausch 453
Fenster 306, 312
Funktionen 30
Kombinationsfenster 550
Listen 583
Parameter 417
Position 185
reagieren auf 315
übergeben 113, 243, 300 f., 306, 325 f.
übersetzen 396
warten auf 405
Zeit 186
Botschaftsfilter
aufrufen 59
Funktionen 593
Botschaftsschlange
erzeugen 346
BringWindowToTop-Funktion 58
BS_XXXX-Konstanten
Pinsel 548 f.

Schalter 548
BuildCommDCB-Funktion 58
BYTE-Typ 518

C
CallMsgFilter-Funktion 59
CallWindowProc-Funktion 59
Carets
anzeigen 381
ausblenden 240
Blinkrate 152, 329
erzeugen 70
freigeben und verwerfen 107
Funktionen 31
Position 152, 330
Catch-Funktion 60
CATCHBUF-Typ 518
CB_ADDSTRING-Botschaft 419
CB_DELETESTRING-Botschaft 420
CB_DIR-Botschaft 420
CB_ERR-Konstante 419 ff., 550
CB_ERRSPACE-Konstante 419 f., 423, 550
CB_FINDSTRING-Botschaft 421
CB_GETCOUNT-Botschaft 421
CB_GETCURSEL-Botschaft 421
CB_GETEDITSEL-Botschaft 422
CB_GETITEMDATA-Botschaft 422
CB_GETLBTEXT-Botschaft 422
CB_GETLBTEXTLEN-Botschaft 423
CB_INSERTSTRING-Botschaft 423
CB_LIMITTEXT-Botschaft 423
CB_RESETCONTENT-Botschaft 424
CB_SELECTSTRING-Botschaft 424
CB_SETCURSEL-Botschaft 425
CB_SETEDITSEL-Botschaft 425
CB_SETITEMDATA-Botschaft 425
CB_SHOWDROPDOWN-Botschaft 426
CB_XXXX-Konstanten 550
CBM_INIT-Konstante 550
CBN_XXXX-Konstanten 550
CBS_DROPDOWN-Konstante 426
CBS_DROPDOWNLIST-Konstante 426
CBS_HASSTRINGS-Konstante 419 ff., 424, 551
CBS_OWNERDRAEFIXED-Konstante 422
CBS_OWNERDRAWFIXED-Konstante 419 ff., 424, 451, 457, 552
CBS_OWNERDRAWVARIABLE-Konstante 419 ff., 424, 451, 457, 552
CBS_SORT-Konstante 451
CBS_XXXX-Konstanten 551
CC_XXXX-Konstanten 552
CCHDEVICENAME-Konstante 555
CE_XXXX-Konstanten 553
CF_OWNERDISPLAY-Konstante 446, 465
CF_TEXT-Konstante 451, 453
CF_XXXX-Konstanten 554
ChangeClipboardChain-Funktion 61
CheckDlgButton-Funktion 61
CheckMenuItem-Funktion 62
CheckRadioButton-Funktion 62
ChildWindowFromPoint-Funktion 63
Chord-Funktion 63
ClearCommBreak-Funktion 64
Client-Fenster 518
CLIENTCREATESTRUCT-Typ 518
ClientToScreen-Funktion 64
CLIP_XXXX-Konstanten 556
ClipCursor-Funktion 65
Clipping-Fähigkeiten 557
Clipping-Regionen
erstellen 136
verschieben 291
CloseClipboard-Funktion 65
CloseComm-Funktion 65
CloseMetaFile-Funktion 66
CloseSound-Funktion 66
CloseWindow-Funktion 66
Code-Segmente
ermitteln 158
Informationen 159
COLOR_XXXX-Konstanten 556
COLORREF-Typ 519
CombineRgn-Funktion 67
COMPAREITEMSTRUCT-Struktur 451
COMPAREITEMSTRUCT-Typ 520
COMSTAT-Typ 520
CopyMetaFile-Funktion 68
CopyRect-Funktion 68
CountClipboardFormats-Funktion 68
CountVoiceNotes-Funktion 68
CP_XXXX-Konstanten 557
CreateBitmap-Funktion 69
CreateBitmapIndirect-Funktion 69
CreateBrushIndirect-Funktion 70
CreateCaret-Funktion 70
CreateCompatibleBitmap-Funktion 71
CreateCompatibleDC-Funktion 72
CreateCursor-Funktion 73
CreateDC-Funktion 73
CreateDialog-Funktion 74 f.
CreateDialogIndirect-Funktion 76
CreateDialogIndirectParam-Funktion 77
CreateDialogParam-Funktion 78
CreateDIBitmap-Funktion 79
CreateDIBPatternBrush-Funktion 80

CreateDiscardableBitmap-Funktion 81
CreateEllipticRgn-Funktion 81
CreateEllipticRgnIndirect-Funktion 82
CreateFont-Funktion 82 f.
CreateFontIndirect-Funktion 84
CreateHatchBrush-Funktion 85
CreateIC-Funktion 85
CreateIcon-Funktion 86
CreateMenu-Funktion 87
CreateMetaFile-Funktion 87
CreatePalette-Funktion 87
CreatePatternBrush-Funktion 87
CreatePen-Funktion 88
CreatePenIndirect-Funktion 89
CreatePolygonRgn-Funktion 89
CreatePolyPolygonRgn-Funktion 90
CreatePopupMenu-Funktion 91
CreateRectRgn-Funktion 91
CreateRectRgnIndirect-Funktion 92
CreateRoundRectRgn-Funktion 92
CreateSolidBrush-Funktion 92
CREATESTRUCT-Struktur 451
CREATESTRUCT-Typ 521
CreateWindow-Funktion 93
CreateWindowEx-Funktion 97
CS_DBLCLKS-Konstante 470, 472
CS_XXXX-Konstanten 557
CTLCOLOR_XXXX-Konstanten 558
CW_USEDEFAULT-Konstante 559

D
Datei-Handle
  verfügbare 343
Dateien
  Dateizeiger 258
  erzeugen 295
  Initialisierung 511
  lesen 277
  löschen 295
  öffnen 255, 275, 295
  schließen 255
  schreiben 279
  temporäre 209
Dateinamen
  Kombinationsfenster und 420
  Listen und 420, 436
  Module 187
Dateiöffnungsflags 595
Datentypen
  BOOL 513
  BYTE 513
  DWORD 513
  LPSTR 513
  WORD 513
DCB-Bit-Flags 522 f.
DCB-Typ 522 f.
DCB_XXXX-Konstanten 559
DDE
  Benachrichtigung 453
DDE-Typen
  DDEAck 513
  DDEAdvise 513
  DDEData 513
  DDEPoke 513
DDEACK-Typ 524
DDEADVISE-Typ 524
DDEDATA-Typ 525
DDEPOKE-Typ 525
DebugBreak-Prozedur 98
Debugger
  Strings senden an 298
DefDlgProc-Funktion 98
DeferWindowPos-Funktion 99
DefFrameProc-Funktion 101
DefHookProc-Funktion 102
DefMDIChildProc-Funktion 103
DefWindowProc-Funktion 104f
DeleteAtom-Funktion 105
DeleteDC-Funktion 105
DELETEITEMSTRUCT-Typ 526
DeleteMenu-Funktion 106
DeleteMetaFile-Funktion 106
DeleteObject-Funktion 106
Desktop
  Handle 165
DestroyCaret-Funktion 107
DestroyCursor-Funktion 107
DestroyIcon-Funktion 108
DestroyMenu-Funktion 108
DestroyWindow-Funktion 108, 458, 460
DEVMODE-Flags 526 f.
DEVMODE-Typ 526 f.
DialogBox-Funktion 109
DialogBoxIndirect-Funktion 110
DialogBoxIndirectParam-Funktion 111
DialogBoxParam-Funktion 112
Dialoge
  Codes 559
  Klassenkonstante 560
  Stile 567
Dialogeintragsvorlagen 528
Dialogfenster
  Basiseinheiten 169
  Botschaften 282 f.
  Botschaftsbearbeitung 98, 252
  Funktionen 31

initialisieren 466
modale 460
modale beenden 124
modale erstellen 109 ff.
nicht-modale erstellen 74 ff.
Schalter 426
Schriften 463, 500
Steuerelementfokus 491
Dialogfenstervorlagen 529
DIB_XXXX-Konstanten 559
DispatchMessage-Funktion 113
DLGC_XXXX-Konstanten 462, 559
DlgDirList-Funktion 113
DlgDirListComboBox-Funktion 115
DlgDirSelect-Funktion 117
DlgDirSelectComboBox-Funktion 117
DLGITEMTEMPLATE-Typ 528
DLGTEMPLATE-Typ 529
DLGTEMPLATEHEADER-Typ 529
DLGWINDOWEXTRA-Konstante 560
DLL
WEP_XXXX-Beendigungscodes 621
DM_GETDEFID-Botschaft 426
DM_SETDEFID-Botschaft 426
DM_XXXX-Konstanten 560
DMBIN_XXXX-Konstanten 561
DMCOLOR_XXXX-Konstanten 561
DMDUP_XXXX-Konstanten 561
DMORIENT_XXXX-Konstanten 562
DMPAPER_XXXX-Konstanten 562
DMRES_XXXX-Konstanten 563
DPtoLP-Funktion 118
DrawFocusRect-Funktion 119
DrawIcon-Funktion 119
DRAWITEMSTRUCT-Struktur 459
DRAWITEMSTRUCT-Typ 530
DrawMenuBar-Funktion 120
DrawText-Funktion 120
DRIVE_XXXX-Konstanten 563
Drucker
Auflösung 563
beidseitig drucken 561
Farben 561
Modus 560
Papierausrichtung 562
Papiereinzüge 561
Papiergrößen 562
Drucker-Escape-Codes 563, 565
DS_LOCALEDIT-Stil 428, 432
DS_XXXX-Konstanten 567
DT_XXXX-Konstanten 568
DWORD-Typ 531
dynamischer Datenaustausch vgl. DDE

E
Editierfelder
Änderungs-Flag 429, 432
Benachrichtigungscodes 569
Bildlauf 431
formatieren 427
formatierendes Rechteck 429, 433
mehrzeilige 428, 430 f., 434
rückgängig machen 435
Stile 570
Tabulatorpositionen setzen 434
Text 428 f., 431 ff., 464
Umbruch 427, 434
Widerrufen-Puffer 427
Zeilen liefern 428
Zeilenanzahl 428
Zeilenindex 430
Zeilenlänge 431
Eigenschaften
auflisten 130 f.
Fenster 198, 314
Funktionen 33
Eingabe
Funktionen 16, 34
Hardware 122
Einheiten
Dialogfenster 281
Ellipse-Funktion 121
Ellipsen
zeichnen 121
EM_CANUNDO-Botschaft 427
EM_EMPTYUNDOBUFFER-Botschaft 427
EM_FMTLINES-Botschaft 427
EM_GETHANDLE-Botschaft 428
EM_GETLINE-Botschaft 428
EM_GETLINECOUNT-Botschaft 428
EM_GETMODIFY-Botschaft 429
EM_GETRECT-Botschaft 429
EM_GETSEL-Botschaft 429
EM_LIMITTEXT-Botschaft 429
EM_LINEFROMCHAR-Botschaft 430
EM_LINEINDEX-Botschaft 430
EM_LINELENGTH-Botschaft 431
EM_LINESCROLL-Botschaft 431
EM_REPLACESEL-Botschaft 431
EM_SETHANDLE-Botschaft 432
EM_SETMODIFY-Botschaft 432
EM_SETPASSWORDCHAR-Botschaft 432
EM_SETRECT-Botschaft 433
EM_SETRECTNP-Botschaft 433
EM_SETSEL-Botschaft 433
EM_SETTABSTOPS-Botschaft 434
EM_SETWORDBREAK-Botschaft 434

EM_UNDO-Botschaft 435
EmptyClipboard-Funktion 121
EN_XXXX-Konstanten 569
EnableHardwareInput-Funktion 122
EnableMenuItem-Funktion 122
EnableWindow-Funktion 123
EndDeferWindowPos-Funktion 123
EndDialog-Funktion 124
EndPaint-Funktion 124
EnumChildWindows-Funktion 125
EnumClipboardFormats-Funktion 126
EnumFonts-Funktion 126 f.
EnumMetaFile-Funktion 128
EnumObjects-Funktion 129
EnumProps-Funktion 130 f.
EnumTaskWindows-Funktion 132
EnumWindows-Funktion 133
EqualRect-Funktion 133
EqualRgn-Funktion 134
Ereignisse
  Schwellenwert 214
  Timer 255
  überprüfen auf 176
erweiterte Fensterstile 625
ES_PASSWORD-Stil 432
ES_XXXX-Konstanten 570 f.
Escape-Funktion 134
Escape-Konstanten für serielle Schnittstellen 572
EscapeCommFunction-Funktion 135
ETO_XXXX-Konstanten 572
EV_XXXX-Konstanten 572
ExcludeClipRect-Funktion 136
ExcludeUpdateRgn-Funktion 136
ExitWindows-Funktion 137
ExtFloodFill-Funktion 137
ExtTextOut-Funktion 138 f.
ExtTextOut-Optionen 572

F
Fähigkeiten
  abrufen 165, 167
  Bogenzeichnung 552
  Clipping 557
  Geräte 575
  Linienzeichnung 585
  Rasterung 600
  Textdarstellung 615
  Vieleckzeichnung 597
Farben 519
  aktualisieren 401
  annähern 188
  auswählen 452
  Bildschirmelemente 203
  Drucker 561
  Hintergrund 150, 327, 546
  palettenrelative 519
  RGB 519, 543 f.
  System 357, 452, 504
  Text 212, 361, 452
Farbtabellen
  Bezeichner 559
FARPROC-Typ 531
FatalExit-Funktion 140
Fehler
  Bearbeitung festlegen 342
  Funktionen 35
  Modus setzen 342
  Schnittstellen 583
  Spooler 609
Feld-Offsets 578
Fenster
  Abmessungen 219 f., 319, 370
  aktives 148
  aktivieren 123, 326, 445, 459
  aktiviert 254
  aktualisieren 402
  Aktualisierungsregion 215 f.
  an oberste Stelle bringen 58
  auflisten 132 f.
  ausblenden 383
  Bildlauf 321, 465
  Bildschirmkontext 218
  blinkende 143
  Botschaften 312, 326
  Client-Bereiche 156, 321, 403 f., 491
  Client-Größe 483
  Desktop 165
  Eigenschaften 314, 349
  Eigenschaften abrufen 198
  enthaltene Punkte 406
  ermitteln 217
  erstellen 93, 95, 97, 451
  Fokus 148, 175, 343, 470, 499
  freigeben und verwerfen 108, 458
  Größe 46, 372 f., 463
  Größe ändern 46 f.
  Handle 533
  initialisieren 451
  Klassen 401, 545
  Klassen-Informationen 153
  Klassennamen 154
  kopieren aus Zwischenablage 494
  Mausbewegungen 484, 499
  Menüs 345
  nächstes 189

Nicht-Client-Bereiche 483
Pop-up 180, 382 f.
Position 372 f.
Punkte in untergeordneten 63
Rahmen 489
schließen 66, 449
sichtbares 254
Standardbotschaftsbearbeitung 104 f.
Symbol 253, 297, 466
system-modal 203, 358
Systemmenüs 204
Systemtasten 503, 506 f.
Tasknummer 221
Text 380, 464
Textlänge 464
übergeordnete 143, 192, 348
Überschrift 221 f., 380
untergeordnete 251, 493
Ursprung 220, 293, 372
vergrößertes 254
verschieben 288, 482
vorhandenes 254
zeichnen 56, 124
zeigen 383, 501
zusätzliche Bytes 219, 222, 330 f., 371, 380, 546
Fenstereigenschaften
auflisten 130 f.
Fenstererstellung
Funktionen 35
Fensterfunktionen
aufrufen 59
Fensterklassen
Informationen 574
registrieren 311
zusätzliche Bytes 154 f.
Fensterstile 622
erweitert 625
FF_XXXX-Konstanten 573
FillRect-Funktion 140
FillRgn-Funktion 141
Filterfunktionen 622
entfernen 398
installieren 374 f., 377, 379
nächste aufrufen 102
FindAtom-Funktion 141
FindResource-Funktion 142
FindWindow-Funktion 143
Flags
Benachrichtigungsfenster 587
Fensterposition setzen 613
FloodFill-Stile 574
globaler Speicher 228, 577
lokaler Speicher 268, 586
Menü 590
neu zeichnen 500
Paletteneinträge 596
Regionen 600
Regionen kombinieren 601
Schnittstellenfehler 553, 583
Schriften 556, 573 f., 596, 607
Schriften abbilden 344
Speicherkonfiguration 223, 621
Steuerelemente 558
Systempalette 614
Textausgabe formatieren 568
Textausrichtung 211, 360
Zeichenabstand 625
Zeichensatz 625
FlashWindow-Funktion 143
FloodFill-Flags 574
FloodFill-Funktion 144
FlushComm-Funktion 145
Fokus
setzen 343
Formate
Anzahl in der Zwischenablage 68
Zwischenablage 193, 252, 311, 446, 554
FrameRect-Funktion 145
FrameRgn-Funktion 146
FreeLibrary-Funktion 146
FreeModule-Funktion 147
FreeProcInstance-Funktion 147
FreeResource-Funktion 147
FreeSelector-Funktion 148
Füllmodi
Polygone 192, 349
Funktionen
Instanzen 147
FW_XXXX-Konstanten 574

G
GCL_XXXX-Konstanten 574
GCW_XXXX-Konstanten 575
GDI vgl. grafische Geräteschnittstelle
Geräte
Fähigkeiten 575
Namen 555
Typen 568
Gerätefähigkeiten
abrufen 165, 167
Gerätefähigkeits-Konstanten 575
Gerätekontexte
erzeugen 72 f.
freigeben 313
Grafikfensterabmessungen 217

Grafikfensterursprung 217
Handle 532
kompatibel 72
löschen 105
Palettenauswahl 324
Speichergerätekontext 72
speichern 318
wiederherstellen 316
Gerätemodi 527
ändern 458
Gerätesteuerblöcke
erstellen 58
GetActiveWindow-Funktion 148
GetAspectRatioFilter-Funktion 148
GetAsyncKeyState-Funktion 149
GetAtomHandle-Funktion 149
GetAtomName-Funktion 149
GetBitmapBits-Funktion 150
GetBitmapDimension-Funktion 150
GetBkColor-Funktion 150
GetBkMode-Funktion 151
GetBrushOrg-Funktion 151
GetBValue-Makro 151
GetCapture-Funktion 152
GetCaretBlinkTime-Funktion 152
GetCaretPos-Funktion 152
GetCharWidth-Funktion 152
GetClassInfo-Funktion 153
GetClassLong-Funktion 154
GetClassName-Funktion 154
GetClassWord-Funktion 155
GetClientRect-Funktion 156
GetClipboardData-Funktion 156
GetClipboardFormatName-Funktion 157
GetClipboardOwner-Funktion 157
GetClipboardViewer-Funktion 158
GetClipBox-Funktion 158
GetCodeHandle-Funktion 158
GetCodeInfo-Funktion 159
GetCommError-Funktion 160
GetCommEventMask-Funktion 161
GetCommState-Funktion 162
GetCurrentPDB-Funktion 162
GetCurrentPosition-Funktion 162
GetCurrentTask-Funktion 163
GetCurrentTime-Funktion 163
GetCursorPos-Funktion 163
GetDC-Funktion 164
GetDCOrg-Funktion 164
GetDesktopWindow-Funktion 165
GetDeviceCaps-Funktion 165, 167
GetDialogBaseUnits-Funktion 169, 434, 444
GetDIBits-Funktion 170
GetDlgCtrlID-Funktion 171
GetDlgItem-Funktion 171
GetDlgItemInt-Funktion 172
GetDlgItemText-Funktion 172
GetDOSEnvironment-Funktion 173
GetDoubleClickTime-Funktion 173
GetDriveType-Funktion 174
GetEnvironment-Funktion 174
GetFocus-Funktion 175
GetFreeSpace-Funktion 175
GetGValue-Makro 176
GetInputState-Funktion 176
GetInstanceData-Funktion 177
GetKBCodePage-Funktion 177
GetKeyboardState-Funktion 178
GetKeyboardType-Funktion 178
GetKeyNameText-Funktion 179
GetKeyState-Funktion 180
GetLastActivePopup-Funktion 180
GetMapMode-Funktion 181
GetMenu-Funktion 181
GetMenuCheckMarkDimensions-Funktion 181
GetMenuItemCount-Funktion 182
GetMenuItemID-Funktion 182
GetMenuState-Funktion 182
GetMenuString-Funktion 183
GetMessage-Funktion 184
GetMessagePos-Funktion 185
GetMessageTime-Funktion 186
GetMetaFile-Funktion 186
GetMetaFileBits-Funktion 186
GetModuleFileName-Funktion 187
GetModuleHandle-Funktion 187
GetModuleUsage-Funktion 188
GetNearestColor-Funktion 188
GetNearestPaletteIndex-Funktion 188
GetNextDlgGroupItem-Funktion 188
GetNextDlgTabItem-Funktion 189
GetNextWindow-Funktion 189
GetNumTasks-Funktion 190, 450
GetObject-Funktion 190
GetPaletteEntries-Funktion 191
GetParent-Funktion 192
GetPixel-Funktion 192
GetPolyFillMode-Funktion 192
GetPriorityClipboardFormat-Funktion 193
GetPrivateProfileInt-Funktion 193
GetPrivateProfileString-Funktion 194
GetProcAddress-Funktion 195
GetProfileInt-Funktion 196

GetProfileString-Funktion 197
GetProp-Funktion 198
GetRgnBox-Funktion 198
GetROP2-Funktion 199
GetRValue-Makro 199
GetScrollPos-Funktion 200
GetScrollRange-Funktion 200
GetStockObject-Funktion 201
GetStretchBltMode-Funktion 202
GetSubMenu-Funktion 203
GetSysColor-Funktion 203
GetSysModalWindow-Funktion 203
GetSystemDirectory-Funktion 204
GetSystemMenu-Funktion 204
GetSystemMetrics-Funktion 205
GetSystemPaletteEntries-Funktion 207
GetSystemPaletteUse-Funktion 207
GetTabbedTextExtent-Funktion 208
GetTempDrive-Funktion 209
GetTempFileName-Funktion 209
GetTextAlign-Funktion 211
GetTextCharacterExtra-Funktion 212
GetTextColor-Funktion 212
GetTextExtent-Funktion 213
GetTextFace-Funktion 213
GetTextMetrics-Funktion 214
GetThresholdEvent-Funktion 214
GetThresholdStatus-Funktion 214
GetTickCount-Funktion 214
GetTopWindow-Funktion 215
GetUpdateRect-Funktion 215
GetUpdateRgn-Funktion 216
GetVersion-Funktion 216
GetViewportExt-Funktion 217
GetViewportOrg-Funktion 217
GetWindow-Funktion 217
GetWindowDC-Funktion 218
GetWindowExt-Funktion 219
GetWindowLong-Funktion 219
GetWindowOrg-Funktion 220
GetWindowRect-Funktion 220
GetWindowsDirectory-Funktion 220
GetWindowTask-Funktion 221
GetWindowText-Funktion 221
GetWindowTextLength-Funktion 222
GetWindowWord-Funktion 222
GetWinFlags-Funktion 223
GlobalAddAtom-Funktion 223
GlobalAlloc-Funktion 224
GlobalCompact-Funktion 225
GlobalDeleteAtom-Funktion 226
GlobalDiscard-Funktion 226
GlobalDosAlloc-Funktion 227
GlobalDosFree-Funktion 227
globale Handle 531
globale Speicherflags 577
globaler Heap
  freier Platz 175
GlobalFindAtom-Funktion 227
GlobalFix-Funktion 228
GlobalFlags-Funktion 228
GlobalFree-Funktion 229
GlobalGetAtomName-Funktion 230
GlobalHandle-Funktion 230
GLOBALHANDLE-Typ 531
GlobalLock-Funktion 230
GlobalLRUNewest-Funktion 231
GlobalLRUOldest-Funktion 231
GlobalNotify-Funktion 232
GlobalPageLock-Funktion 233
GlobalPageUnlock-Funktion 233
GlobalReAlloc-Funktion 234
GlobalSize-Funktion 235
GlobalUnfix-Funktion 236
GlobalUnlock-Funktion 236
GlobalUnWire-Funktion 237
GlobalWire-Funktion 237
GMEM_XXXX-Konstanten 577
Grafikfenster
  Abmessungen 217, 318, 364
  Ursprung 217, 292, 365
grafische Geräteschnittstelle 164, 323, 328
  Aufrufe 128
  COLORREF-Typ 519
  CreateFont-Funktion 82
  Drucker-Escape-Codes 563
  Escape-Funktion 134
  GetWindow-Konstanten 578
  Klassentyp-Konstanten 593
  META_ARC-Code 589
  Metafile-Struktur 303
  UnrealizeObject-Funktion 400
GrayString-Funktion 238
Gruppen
  Steuerelemente und 188
GW_XXXX-Konstanten 578
GWL_XXXX-Konstanten 578
GWW_XXXX-Konstanten 579

H
Handle
  Bitmap 532
  Fenster 533
  Gerätekontexte 532
  globale 531

lokaler Speicher 534
Mauszeiger 532
Menüoptionsvorlage 538
Menüs 533
Metafiles 539
Paletten 533
Pinsel 532
Regionen 533
Schriften 533
Stifte 533
Strings 533
Symbole 533
Handle-Tabelle 532
HANDLE-Typ 532
Handle-Typen
GLOBALHANDLE 513
HBITMAP 513
HBRUSH 513
HCURSOR 513
HDC 513
HFONT 513
HICON 513
HMENU 513
HPALETTE 513
HPEN 513
HRGN 513
HSTR 513
HWND 513
HANDLETABLE-Typ 532
Hardware
Funktionen 37
HBITMAP-Typ 532
HBRUSH-Typ 532
HCURSOR-Typ 532
HDC-Typ 532
Header
Bitmap-Core 514
Bitmap-Dateien 516
Bitmap-Informationen 517
Heap
globaler 224 ff.
lokaler 269 ff.
HELP_XXXX-Konstanten 579
HFONT-Typ 533
HIBYTE-Makro 240
HICON-Typ 533
HideCaret-Funktion 240
HIDEWINDOW-Konstante 607
Hilfesystem
aufrufen 408
HiliteMenuItem-Funktion 241
Hintergrund
Farben 150, 327
löschen 461
Modus 151, 328
neu zeichnen 461
Hintergrundmodi-Konstanten 580
HIWORD-Makro 242
HMENU-Typ 533
Höhen-/Seitenverhältnis 148
HPALETTE-Typ 533
HPEN-Typ 533
HRGN-Typ 533
HS_XXXX-Konstanten 580
HSTR-Typ 533
HTXXXX-Konstanten 581
Hüllkurve
Klangwarteschlange 367
HWND-Typ 533

I
ID_XXXX-Konstanten 582
IDC_XXXX-Konstanten 582
IDI_XXXX-Konstanten 583
IDs
Mauszeiger 582
Menüoptionen 182
Symbole 583
IE_XXXX-Konstanten 583
Index
Paletten 298
InflateRect-Funktion 242
Information
Funktionen 37
Informationskontexte
erstellen 85
InitAtomTable-Funktion 242
Initialisierungsdateien
ändern 511
private 193 f.
schreiben in 410
Windows 196, 411
InSendMessage-Funktion 243
InsertMenu-Funktion 243, 245
Instanzdaten
kopieren 177
Integer
Zeiger 543
IntersectClipRect-Funktion 246
IntersectRect-Funktion 247
InvalidateRect-Funktion 247
InvalidateRgn-Funktion 248
InvertRect-Funktion 249
InvertRgn-Funktion 250
IsCharAlpha-Funktion 250
IsCharAlphaNumeric-Funktion 250

IsCharLower-Funktion 251
IsCharUpper-Funktion 251
IsChild-Funktion 251
IsClipboardFormatAvailable-Funktion 252
IsDialogMessage-Funktion 252
IsDlgButtonChecked-Funktion 253
IsIconic-Funktion 253
IsRectEmpty-Funktion 253
IsWindow-Funktion 254
IsWindowEnabled-Funktion 254
IsWindowVisible-Funktion 254
IsZoomed-Funktion 254

K
KillTimer-Funktion 255
Klangausgabegerät
  öffnen 297
Klänge
  Ausgabegerät öffnen 297
  beenden 66, 385
  setzen 355
  Warnton 282
  warten auf Status 406
  Warteschlange 369
  Wiedergabe 384
Klangkonstanten 602
Klangwarteschlange
  abspielen 384
  Anzahl Noten 68
  beenden 385
  Größe festlegen 368
  Hüllkurve 367
  leeren 385
  Note hinzufügen 368
  Schwellenwert 369
  synchronisieren 390
  Ton hinzufügen 369
  Ton setzen 366
Klangwarteschlangen
  leeren 66
Klassen
  Bezeichnungen 546
  Fenster 546
  Hintergrundfarben 546
  Informationen abrufen 153
  Mauszeiger 546
  Menüs 546
  Namen 545
  Namen abrufen 154
  registrieren 311
  Registrierung 401
  Symbole 546
  virtuelle 620
  WNDCLASS-Typ 545
  zusätzliche Bytes 154 f.
Kombinationsfenster
  aktuelle Auswahl 117
  ausklappende Listen 426
  Benachrichtigungscodes 550
  besitzergezeichnete Einträge 451
  Botschaften 550
  Dateinamen und 420
  Einträge 424 f., 457
  Größe 479
  Stile 551
  Strings 419 ff.
  Text 422, 424 f., 464
  Verzeichnislisten in 115
Kompressionskonstanten
  Bitmaps 547
Konstanten
  Bildlaufleisten 603
  CreateWindow 575
  Dateien öffnen 595
  Dialogfenster 559
  Drucker-Escape 563
  Escape 572
  Filterverwaltung 622
  GetWindow-Funktion 578
  Hintergrundmodi 580
  Klänge 602
  Metafile 589
  Schnittstellenereignisse 572
  ShowWindow 612
  Steuerelemente 558
  Systemfarben 556
  WM_SHOWWINDOW-Botschaft 613
Koordinaten
  aktuelle Position 162, 288
  Bildschirm 64, 319
  Client 64, 156, 319
  konvertieren 64, 319
  Punkte 543
  Rechtecke 350
  Ursprung des Client-Bereichs 164

L
Laufwerke
  temporäre Dateien 209
  Typbestimmung 174
Laufwerkstypen 563
Lautsprecher 66
LB_ADDSTRING-Botschaft 436
LB_DELETESTRING-Botschaft 436, 457
LB_DIR-Botschaft 436

LB_ERR-Konstante 436 ff., 464
LB_ERRSPACE-Konstante 436, 440
LB_FINDSTRING-Botschaft 437
LB_GETCOUNT-Botschaft 437
LB_GETCURSEL-Botschaft 438
LB_GETHORIZONTALEXTENT-Botschaft 438
LB_GETITEMDATA-Botschaft 438
LB_GETITEMRECT-Botschaft 438
LB_GETSEL-Botschaft 439
LB_GETSELCOUNT-Botschaft 439
LB_GETSELITEMS-Botschaft 439
LB_GETTEXT-Botschaft 440
LB_GETTEXTLEN-Botschaft 440
LB_GETTOPINDEX-Botschaft 440
LB_INSERTSTRING-Botschaft 440
LB_RESETCONTENT-Botschaft 441
LB_SELECTSTRING-Botschaft 441
LB_SELITEMRANGE-Botschaft 442
LB_SETCOLUMNWIDTH-Botschaft 442
LB_SETCURSEL-Botschaft 442
LB_SETHORIZONTALEXTENT-Botschaft 443
LB_SETITEMDATA-Botschaft 443
LB_SETSEL-Botschaft 444
LB_SETTABSTOPS-Botschaft 444
LB_SETTOPINDEX-Botschaft 445
LB_XXXX-Konstanten 583
LBN_XXXX-Konstanten 584
LBS_HASSTRINGS-Stil 436 f., 440 f., 584
LBS_MULTICOLUMN-Stil 442
LBS_OWNERDRAWFIXED-Stil 436 f., 440 f., 451, 457, 585
LBS_OWNERDRAWVARIABLE-Stil 436 f., 440 f., 451, 457, 585
LBS_SORT-Stil 451, 585
LBS_WANTKEYBOARDINPUT-Stil 448
LBS_XXXX-Konstanten 584
LC_XXXX-Konstanten 585
_lclose-Funktion 255
_lcreat-Funktion 255
Leerraum
  Zeichen 361
  zusätzlicher 212
Leistungsmerkmale
  Geräte 575
LF_FACESIZE-Konstante 586
LimitEmsPages-Funktion 256
LineDDA-Funktion 256
LineTo-Funktion 257
Linien
  zeichnen 257
Linienzeichnungsfähigkeiten 585
Listen
  aktuelle Auswahl 117
  Beantwortung virtueller Tasten 510
  Benachrichtigungscodes 584
  besitzergezeichnete Einträge 451
  Bildlaufbereich 438, 443
  Dateinamen und 436
  Einträge 439 ff., 444, 457
  Größe 479
  Rahmen 438
  Rückgabewerte 583
  Spaltenbreite 442
  Stile 584
  Strings 436 ff.
  Tabulatoren 444
  Tastendrücke und 447
  Verzeichnislisten 113
_llseek-Funktion 258
LMEM_XXXX-Konstanten 586
LoadAccelerators-Funktion 258
LoadBitmap-Funktion 259
LoadCursor-Funktion 260
LoadIcon-Funktion 261
LoadLibrary-Funktion 262
LoadMenu-Funktion 262
LoadMenuIndirect-Funktion 263
LoadModule-Funktion 263
LoadResource-Funktion 265
LoadString-Funktion 266
LOBYTE-Makro 266
LocalAlloc-Funktion 266
LocalCompact-Funktion 267
LocalDiscard-Funktion 268
LocalFlags-Funktion 268
LocalFree-Funktion 269
LocalHandle-Funktion 269
LOCALHANDLE-Typ 534
LocalInit-Funktion 269
LocalLock-Funktion 270
LocalReAlloc-Funktion 270 f.
LocalShrink-Funktion 272
LocalSize-Funktion 272
LocalUnlock-Funktion 273
LockData-Makro 273
LockResource-Funktion 273
LockSegment-Funktion 274
LOGBRUSH-Stile 534
LOGBRUSH-Typ 534
LOGFONT-Typ 535
LOGPALETTE-Typ 536
LOGPEN-Typ 536
lokale Speicher-Flags 586
lokales Handle 534

_lopen-Funktion 275
LOWORD-Makro 276
LPtoDP-Funktion 276
LPTx-Konstante 587
_lread-Funktion 277
lstrcat-Funktion 277
lstrcmp-Funktion 278
lstrcmpi-Funktion 278
lstrcpy-Funktion 279
lstrlen-Funktion 279
_lwrite-Funktion 279

M
MA_XXXX-Konstanten 587
MAKELONG-Makro 280
MakeProcInstance-Funktion 280, 434
Makros
  GetBValue 151
  GetGValue 176
  GetRValue 199
  HIBYTE 240
  HIWORD 242
  LOBYTE 266
  LockData 273
  LOWORD 276
  MAKELONG 280
  PaletteIndex 298
  PaletteRGB 299
  RGB 317
  UnlockData 399
  UnlockResource 399
MapDialogRect-Funktion 281
MapVirtualKey-Funktion 281
Maße
  System 205
  Text 544
Masken
  Tastenstatus 592
Maus
  aktivieren 587
  Bewegungen 481, 484, 488
  Doppelklick 341, 472
  Doppelklick-Intervall 173
  Fenster aktivieren 481
  linke Taste 471, 485 f.
  mittlere Taste 472f, 486 ff.
  rechte Taste 489 f., 496 f.
  Tasten 388
Mausfang 152
  aufheben 312
  setzen 329
Mauszeiger
  aktuelle Position 163
  beschränken 65
  Bewegungen 499
  Clipping 65
  erstellen 73
  Form 336
  freigeben und verwerfen 107
  Funktionen 38
  Handle 532
  IDs 582
  Klassen 546
  laden 260
  Position 185, 337
  zeigen 381
Mauszeigerpositionscodes 581
MB_XXXX-Konstanten 587
MDI
  Symbole anordnen 476
  untergeordnete Fenster 475 f.
MDICREATESTRUCT-Typ 536
MEASUREITEMSTRUCT-Typ 537
Meldungsfenster 446
  erstellen 282 f.
Menüeintragsvorlagen 538
MENUITEMTEMPLATE-Typ 538
MENUITEMTEMPLATEHEADER-Typ 538
Menüleisten
  zeichnen 120
Menüoptionen
  aktivieren 122
  ändern 285, 287
  Anzahl 182
  auswählen 480
  Bitmaps als 345
  einfügen 243, 245
  Header 538
  hervorheben 241
  IDs 182
  Label 183
  löschen 314
Menüs 263, 460
  abrufen 181
  Auswahlmarkierungen 181
  erzeugen 87
  Fehlermeldungen 479
  Flags 590
  freigeben und verwerfen 108
  Funktionen 39
  Größe 479
  Handle 181, 533
  initialisieren 467
  Klassen 546
  laden 262
  löschen 106

Optionen anfügen 53
Optionen markieren 62
Pop-up 91, 203, 394, 467
Pop-up ermitteln 53
setzen 345
Status abfragen 182
System 204
MessageBeep-Funktion 282
MessageBox-Funktion 282 f.
META_XXXX-Konstanten 589
METAFILEPICT-Typ 538
Metafiles
abrufen 186
auflisten 128
ausführen 303
Bits setzen 347
erzeugen 87
Formate 538
Größe von 186
Header 539
Konstanten 589
kopieren 68
löschen 106
Records 539
schließen 66
Strukturen 303, 539
METAHEADER-Typ 539
METARECORD-Typ 539
MF_XXXX-Konstanten 590 f.
Millisekunden
seit Systemstart 214
MK_XXXX-Konstanten 592
MM_TEXT-Konstante 461
MM_XXXX-Konstanten 592
Modi
Abbildung 181, 344, 592
aufheben 446
Hintergrund 151, 327 f., 580
Polygone 598
Polygone füllen 192
strecken 202, 355, 612
zeichnen 352
ModifyMenu-Funktion 285, 287
Module
ausführbare Dateien 187
Bibliotheken 262
freigeben 147
Handles 187
laden 263
Referenzzähler abfragen 188
MoveTo-Funktion 288
MoveWindow-Funktion 288
MSG-Typ 540
MSGF_DIALOGBOX-Konstante 460
MSGF_MENU-Konstante 460
MSGF_XXXX-Konstanten 593
MulDiv-Funktion 289
MULTIKEYHELP-Typ 541
Muster 542

N
Namen
Klassen 546
Noten
Anzahl in Klangwarteschlange 68

O
OBJ_XXXX-Konstanten 593
Objekte
auflisten 129
auswählen 323
laden 190
löschen 106
virtuelle 620
vordefinierte 201
OBM_XXXX-Konstanten 593
ODA_XXXX-Konstanten 594
ODS_XXXX-Konstanten 594
ODT_XXXX-Konstanten 594
OemKeyScan-Funktion 290
OemToAnsi-Funktion 290 f.
OemToAnsiBuff-Funktion 291
OF_XXXX-Konstanten 595
OffsetClipRgn-Funktion 291
OffsetRect-Funktion 291
OffsetRgn-Funktion 292
Offsets 579
Klassenfelder 574
OffsetViewportOrg-Funktion 292
OffsetWindowOrg-Funktion 293
OFSTRUCT-Typ 541
OpenClipboard-Funktion 293
OpenComm-Funktion 294
OpenFile-Funktion 295
OpenIcon-Funktion 297
OpenSound-Funktion 297
Optionen
ExtTextOut-Funktion 572
PeekMessage-Funktion 598
Textausrichtung 614
OUT_XXXX-Konstanten 596
OutputDebugString-Funktion 298

P
PaintRgn-Funktion 298
PAINTSTRUCT-Typ 541

Palette
  Einträge ändern 49
  System 207, 358 f.
PALETTEENTRY-Typ 542
PaletteIndex-Makro 298
Paletten
  auswählen 324
  Einträge 207, 347, 542, 596
  Einträge abrufen 191
  erstellen 87
  Größe ändern 316
  Handle 533
  Index 520
  löschen 106
  realisieren 309
  System 207, 309, 493
  virtuelle 298, 309, 495, 536
PaletteRGB-Makro 299
Papiergrößen 562
PatBlt-Funktion 299
PATTERN-Typ 542
PC_XXXX-Konstanten 596 f.
PDB (Program Data Base)
  abrufen 162
PeekMessage-Funktion 300 f.
PeekMessage-Optionen 598
Pie-Funktion 302
Pinsel
  auflisten 593
  erstellen 92
  erstellen aus Bitmaps 80
  erzeugen 70, 85, 87
  Handle 201, 532
  löschen 106
  Stile 548
  Ursprung 151, 328
  virtuelle 534
  vordefinierte 201
PINT-Typ 543
Pixel
  Farbe ermitteln 192
  setzen 348
PlayMetaFile-Funktion 303
PlayMetaFileRecord-Funktion 303
PM_XXXX-Konstanten 598
POINT-Typ 543
Polygon-Funktion 304
Polygone
  Füllmodus 192, 305, 349
  mehrere 305
  zeichnen 304
Polygonfüllmodi-Konstanten 598
Polyline-Funktion 304
PolyPolygon-Funktion 305
Position
  aktuelle 288
  aktuelle Koordinaten 162
  Bildlaufleisten 200, 353
  Fenster 372 f.
  Mauszeiger 163, 185
PostAppMessage-Funktion 305
PostMessage-Funktion 306
PostQuitMessage-Funktion 307, 460
PR_JOBSTATUS-Konstante 598
PROC_XXXX-Konstanten 598
Prozeduren
  far 531
PS_XXXX-Konstanten 599
PtInRect-Funktion 307
PtInRegion-Funktion 307
PtVisible-Funktion 308
Puffer
  an Schnittstelle schreiben 409
  Catch-Funktion 518
Punkte
  berechnen 256
  in Fenstern enthalten 406
  in Rechtecken 307
  in Regionen 307
  in untergeordneten Fenstern 63
  konvertieren 118, 276
  sichtbar 308

Q
Quit-Botschaft
  übergeben 307

R
R2_XXXX-Konstanten 599
Rahmenfenster
  Botschaftsbearbeitung 101
  Rechteck 145
  Region 146
Raster
  binäre Operationen 599
  Fähigkeiten 600
  ternäre Operationen 615
Rasteroperationen
  binäre 599
  ternäre 615
RC_XXXX-Konstanten 600
ReadComm-Funktion 308
RealizePalette-Funktion 309
Rechtecke 543
  abgerundete 92
  abrunden 317

Aktualisierungsregion 215
ausfüllen 140
Clipping 158, 246
definieren für Clipping 136
erstellen aus Regionen 351
Farben invertieren 249
Fokus-Stil 119
Funktionen 9
Gültigkeit überprüfen 403
Koordinaten 350
kopieren 68
leere 253, 351
Punkte innerhalb 307
Rahmen zeichnen 145
sichtbar 310
Überschneidungsbereich 247
umgebende Regionen 198
umgrenzende 220
ungültig machen 247
vereinigen 399
vergleichen 133
vergrößern 242
verschieben 291
zeichnen 309
Records
Metafiles 539
RECT-Typ 543
Rectangle-Funktion 309
RectInRegion-Funktion 310
RectVisible-Funktion 310
Referenzzähler
abfragen 188
Region-Flags 600
Regionen
aktualisieren 216
Aktualisierung 215
ausfüllen 137, 141
Clipping 246, 291, 308
definieren für Clipping 136
elliptische 81
Farben invertieren 250
Gültigkeit prüfen 404
Handle 533
kombinieren 67
leere 351
löschen 106
Polygon 89
Polygone erstellen 90
Punkte innerhalb von 307
Rahmen zeichnen 146
Rechtecke erzeugen aus 351
rechteckige 91 f., 198
ungültig machen 248
vergleichen 134
verschieben 292
zeichnen 298
RegisterClass-Funktion 311
RegisterClipboardFormat-Funktion 311
RegisterWindowMessage-Funktion 312
ReleaseCapture-Funktion 312
ReleaseDC-Funktion 313
RemoveFontResource-Funktion 313
RemoveMenu-Funktion 314
RemoveProp-Funktion 314
ReplyMessage-Funktion 315
ResizePalette-Funktion 316
Ressourcen
auffinden 142
Bitmaps laden 259
Callback-Funktion 351
freigeben 147
Größe von 384
laden 265
Mauszeiger 260
Menüs 262
Schriften 461
Speicher reservieren 48
sperren 273
Strings laden 266
Symbole laden 261
Typen 601
Zugriff auf 44
RestoreDC-Funktion 316
RGB-Farben 519, 543 f.
RGB-Makro 317
RGBQUAD-Typ 543
RGBTRIPLE-Typ 544
RGN_XXXX-Konstanten 601
RoundRect-Funktion 317
RT_XXXX-Konstanten 601

S

S_XXXX-Konstanten 602
SaveDC-Funktion 318
SB_XXXX-Konstanten 603
SBS_XXXX-Konstanten 604
SC_XXXX-Konstanten 605
ScaleViewportExt-Funktion 318
ScaleWindowExt-Funktion 319
Schalter
auswählen 61
Benachrichtigungscodes 547
Status 417 f.
Stile 419, 548
Text 464
Voreinstellung 426

Schaltfelder
markieren 62, 418
Status 417
Schnittstellen 64
empfangen 398
Ereigniskonstanten 572
erweiterte Funktionen 135
Escape-Konstanten 572
Fehler 553
Konfigurationskonstanten 606
lesen 308
öffnen 294
schließen 65
Status 336, 520
Unterbrechungen 64
Warteschlange leeren 145
Zeichen übertragen 397
Schnittstellenfehlerflags 583
Schnittstellengeräte
Ereignismaske leeren 161
Ereignismasken 335
Ereignismasken abrufen 161
Fehler 160
Gerätesteuerblock abrufen 162
Puffer ausgeben 409
reinitialisieren 336
Unterbrechung 335
Zeichen 398
Schriftbild
aktuelles 213
Schriftdarstellungsqualitäts-Konstanten 607
Schriften
abbilden 344
Abmessungen 214
ändern 461
auflisten 126 f.
Ausgabepräzisions-Flags 596
Ausgabequalität 607
Clipping-Präzisions-Flags 556
Dialogfenster und 463
erzeugen 82 f.
Familien-Flags 573
Handle 533
löschen 106
Namen 586
Ressourcen 45, 313, 461
Ressourcen hinzufügen 44
Schriftbild 213
Schriftstärken-Flags 574
setzen 500
virtuelle 535
vordefinierte 201
Zeichenabstand 625
Zeichensatz-Flags 625
Schwellenwert
Ereignisse 214
Status 214
ScreenToClient-Funktion 319
ScrollDC-Funktion 320
ScrollWindow-Funktion 321
SelectClipRgn-Funktion 322
SelectObject-Funktion 323
SelectPalette-Funktion 324
Selektoren
freigeben 148
SendDlgItemMessage-Funktion 325
SendMessage-Funktion 326, 447, 453, 459
SetActiveWindow-Funktion 326
SetBitmapBits-Funktion 327
SetBitmapDimension-Funktion 327
SetBkColor-Funktion 327
SetBkMode-Funktion 328
SetBrushOrg-Funktion 328
SetCapture-Funktion 329
SetCaretBlinkTime-Funktion 329
SetCaretPos-Funktion 330
SetClassLong-Funktion 330
SetClassWord-Funktion 331
SetClipboardData-Funktion 332 f., 446, 465
SetClipboardViewer-Funktion 334, 447, 459
SetCommBreak-Funktion 335
SetCommEventMask-Funktion 335
SetCommState-Funktion 336
SetCursor-Funktion 336
SetCursorPos-Funktion 337
SetDIBits-Funktion 338
SetDIBitsToDevice-Funktion 339
SetDlgItemInt-Funktion 340
SetDlgItemText-Funktion 340
SetDoubleClickTime-Funktion 341
SetEnvironment-Funktion 341
SetErrorMode-Funktion 342
SetFocus-Funktion 343
SetHandleCount-Funktion 343
SetKeyboardState-Funktion 343
SetMapMode-Funktion 344
SetMapperFlags-Funktion 344
SetMenu-Funktion 345
SetMenuItemBitmaps-Funktion 345
SetMessageQueue-Funktion 346
SetMetaFileBits-Funktion 347
SetPaletteEntries-Funktion 347

SetParent-Funktion 348
SetPixel-Funktion 348
SetPolyFillMode-Funktion 349
SetProp-Funktion 349
SetRect-Funktion 350
SetRectEmpty-Funktion 351
SetRectRgn-Funktion 351
SetResourceHandler-Funktion 351
SetROP2-Funktion 352
SetScrollPos-Funktion 353, 465
SetScrollRange-Funktion 354
SetSoundNoise-Funktion 355
SetStretchBltMode-Funktion 355
SetSwapAreaSize-Funktion 356
SetSysColors-Funktion 357
SetSysModalWindow-Funktion 358
SetSystemPaletteUse-Funktion 358 f.
SetTextAlign-Funktion 360
SetTextCharacterExtra-Funktion 361
SetTextColor-Funktion 361
SetTextJustification-Funktion 362
SetTimer-Funktion 363
SetViewportExt-Funktion 364
SetViewportOrg-Funktion 365
SetVoiceAccent-Funktion 366
SetVoiceEnvelope-Funktion 367
SetVoiceNote-Funktion 368
SetVoiceQueueSize-Funktion 368
SetVoiceSound-Funktion 369
SetVoiceThreshold-Funktion 369
SetWindowExt-Funktion 370
SetWindowLong-Funktion 371
SetWindowOrg-Funktion 372
SetWindowPos-Flags 613
SetWindowPos-Funktion 372f
SetWindowsHook-Funktion 374 ff.
SetWindowText-Funktion 380
SetWindowWord-Funktion 380
ShowCaret-Funktion 381
ShowCursor-Funktion 381
ShowOwnedPopups-Funktion 382
ShowScrollBar-Funktion 382
ShowWindow-Botschaftskonstanten 613
ShowWindow-Funktion 383
ShowWindow-Konstanten 612
SizeOfResource-Funktion 384
SIZEXXXX-Konstanten 607
SM_XXXX-Konstanten 608
SP_XXXX-Konstanten 609
Speicher
  Expanded 256
  globaler 226, 228 ff., 234 ff.
  lokaler 266 ff.
  Segmente 273f, 400, 402
  Swap-Größe setzen 356
  verdichten 450
Speicherkonfiguration-Flags 223, 621
Spooler
  Fehlercodes 609
  Status-Code 598
Spooler-Status 503
SS_XXXX-Konstanten 610 f.
StartSound-Funktion 384
statische Steuerelemente
  Stile 610
Status
  Schwellenwert 214
Steuerelemente
  ausgewählte Schalter 253
  besitzergezeichnete 459
  Botschaften 325
  Eingabefokus 466
  gruppierte 188
  Handle 171
  IDs 171
  Tabulatorstopps 189
  Text 172, 221 f., 340
Stifte
  erzeugen 88
  Handle 533
  löschen 106
  Stile 599
  virtuelle 536
  vordefinierte 201
Stile
  Bildlaufleisten 604
  Dialoge 567
  Editierfelder 570
  Fenster 622, 625
  Klassen 546
  Kombinationsfenster 551
  Listen 584
  Pinsel 548
  Pinselmuster 580
  Schalter 419, 548
  statische Steuerelemente 610
  Stifte 599
StopSound-Funktion 385
Streckmodus
  aktueller 202
  setzen 355
StretchBlt-Funktion 385
StretchBlt-Modi 612
StretchDIBits-Funktion 387
Strings
  Abmessungen 208, 213

an Debugger senden 298
Ausgabe 238 f.
Handle 533
konkatenieren 277
kopieren 279
laden 266
Länge 279
schreiben 138 f.
umwandeln in Großbuchstaben 52
umwandeln in Kleinbuchstaben 50
vergleichen 278
Zeichensätze konvertieren 51 f.
Strukturen
Metafiles 539
SW_XXXX-Botschaftskonstanten 613
SW_XXXX-Konstanten 612
Swap-Speichergröße setzen 356
SwapMouseButton-Funktion 388
SwapRecording-Funktion 389
SwitchStackBack-Funktion 389
SwitchStackTo-Funktion 389
SWP_XXXX-Konstanten 613
Symbole
anordnen 55
erzeugen 86
Fenster 66
freigeben und verwerfen 108
Handle 533
Hintergrund 466
Klasse 546
laden 261
malen 492
neu zeichnen 466
öffnen 297
Standard-IDs 583
Status prüfen 253
verschieben 494
zeichnen 119
SyncAllVoices-Funktion 390
SYSPAL_XXXX-Konstanten 614
System
Funktionen 40
system-modales Fenster 203, 358
System-Warteschlange
überprüfen 176
Systembefehlswerte 605
Systemfarben
ändern 504
Codes 556
setzen 357
Systemmaße
abrufen 205
Codes 608
Systemmenüoptionen
vergrößern 505
verkleinern 505
Systempalette
ändern 493
Einträge 207
Flags 614
Verwendung 358 f.
Zugriff 207
Systemregister
Werte 60
Systemstartzeit 214
Systemstillstand 460
Systemverzeichnis
Name 204
Systemzeit
ändern 509

T

TA_XXXX-Konstanten 614
TabbedTextOut-Funktion 391
Tabulatorpositionen
Editierfelder und 434
in Listen 444
Tasks
aktuelle 163
Anzahl 190, 221
auflisten 132
Handle 163
unterbrechen 415
Tastatur
Botschaften in Zeichen übersetzen 396
Code-Seite ermitteln 177
Status 178, 343
Typ ermitteln 178
Tasten
gedrückt 468
losgelassene 469
Namen ermitteln 179
Standard 468 f.
Status ermitteln 180
Tastendrücke 447 f.
tote Taste 456
Tastenstatusmasken 592
TC_XXXX-Konstanten 615
temporäre Dateien
Laufwerke 209, 616
Namen 209
ternäre Rasteroperationen 615
Text
Abmessungen 208
Ausgabe 391
Ausrichtung 362

Ausrichtungsflags 211, 360
Ausrichtungsoptionen 614
ausschneiden 453
Darstellungsfähigkeiten 615
Farbe 212, 361
Farben 452
Fenster 221 f., 464
formatierte Ausgabe 120
Formatierungsflags 568
kopieren 451
Länge 464
löschen 449
Maße 544
setzen 501
zeichnen 392
zusätzlicher Leerraum 212
TEXTMETRIC-Typ 544
TextOut-Funktion 392
TF_FORCEDRIVE-Konstante 616
Throw-Funktion 392
Thunk 389
Timer
Ereignisse 255
setzen 363
Zeitbegrenzung 509
ToAscii-Funktion 393
Tortendiagramme
zeichnen 302
TrackPopupMenu-Funktion 394
TranslateAccelerator-Funktion 395
TranslateMDISysAccel-Funktion 396
TranslateMessage-Funktion 396
TransmitCommChar-Funktion 397
Typen
Geräte 568
Ressourcen 601

U
übergeordnete Fenster
abrufen 192
ändern 348
ermitteln 143
Umgebung
ersetzen 341
erstellen 341
Systemanschluß 174
UngetCommChar-Funktion 398
UnhookWindowsHook-Funktion 398
UnionRect-Funktion 399
UnlockData-Makro 399
UnlockResource-Makro 399
UnlockSegment-Funktion 400
UnrealizeObject-Funktion 400
UnregisterClass-Funktion 401
untergeordnete Fenster
aktivieren 449
auflisten 125
bearbeiten 493
Botschaftsbearbeitung 103
Ebenen 215
enthaltene Punkte 63
MDI 103, 474
Symbole anordnen 476
UpdateColors-Funktion 401
UpdateWindow-Funktion 402
Ursprung
Grafikfenster 217

V
ValidateCodeSegments-Funktion 402
ValidateFreeSpaces-Funktion 403
ValidateRect-Funktion 403
ValidateRgn-Funktion 404
Version
von Windows 216
Verzeichnis
System 204
Windows 220
Verzeichnislisten
in Kombinationsfenstern 115
Verzeichnisse
Namen in Listen 113
Vielecke
siehe auch Polygone 304
Vieleckzeichnungsfähigkeiten 597
virtuelle Tasten
abbilden 281
Codes 616 ff.
Listen 510
Status 149
Zeichen konvertieren 404
VK_XXXX-Konstanten 616 ff.
VkKeyScan-Funktion 404
vordefinierte virtuelle Klassen 620

W
WaitMessage-Funktion 405
WaitSoundState-Funktion 406
Warnton 282
WEP_XXXX-Konstanten 621
WF_XXXX-Konstanten 621
WH_XXXX-Konstanten 622
widerrufen 509
WindowFromPoint-Funktion 406
Windows
beenden 137

Filterfunktionscodes 622
Speicherkonfigurationsflags 621
Version 216
Verzeichnis 220
WinExec-Funktion 406 f.
WinHelp-Funktion 408
WM_ACTIVATE-Botschaft 445
WM_ACTIVATEAPP-Botschaft 445
WM_ASKCBFORMATNAME-Botschaft 446
WM_CANCELMODE-Botschaft 446
WM_CHANGECBCHAIN-Botschaft 447, 459
WM_CHAR-Botschaft 447
WM_CHARTOITEM-Botschaft 448
WM_CHILDACTIVATE-Botschaft 449
WM_CLEAR-Botschaft 449
WM_CLOSE-Botschaft 449
WM_COMMAND-Botschaft 449
WM_COMPACTING-Botschaft 450
WM_COMPAREITEM-Botschaft 419, 436, 451
WM_COPY-Botschaft 451
WM_CREATE-Botschaft 451
WM_CTLCOLOR-Botschaft 452
WM_CUT-Botschaft 453
WM_DDE_ACK-Botschaft 453
WM_DDE_ADVISE-Botschaft 453
WM_DDE_DATA-Botschaft 454
WM_DDE_EXECUTE-Botschaft 453 f.
WM_DDE_INITIATE-Botschaft 454
WM_DDE_POKE-Botschaft 455
WM_DDE_REQUEST-Botschaft 455
WM_DDE_TERMINATE-Botschaft 456
WM_DDE_UNADVISE-Botschaft 456
WM_DEADCHAR-Botschaft 456
WM_DELETEITEM-Botschaft 420, 424, 436, 457
WM_DESTROY-Botschaft 447, 458 f.
WM_DESTROYCLIPBOARD-Botschaft 458
WM_DEVMODECHANGE-Botschaft 458
WM_DRAWCLIPBOARD-Botschaft 459
WM_DRAWITEM-Botschaft 459
WM_ENABLE-Botschaft 459
WM_ENDSESSION-Botschaft 460
WM_ENTERIDLE-Botschaft 460
WM_ERASEBKGND-Botschaft 461, 466
WM_FONTCHANGE-Botschaft 461
WM_GETDLGCODE-Botschaft 462
WM_GETFONT-Botschaft 463
WM_GETMINMAXINFO-Botschaft 463
WM_GETTEXT-Botschaft 464
WM_GETTEXTLENGTH-Botschaft 464
WM_HSCROLL-Botschaft 465
WM_HSCROLLCLIPBOARD-Botschaft 465
WM_ICONERASEBKGND-Botschaft 466
WM_INITDIALOG-Botschaft 466
WM_INITMENU-Botschaft 467
WM_INITMENUPOPUP-Botschaft 467
WM_KEYDOWN-Botschaft 447, 468
WM_KEYUP-Botschaft 469
WM_KILLFOCUS-Botschaft 470
WM_LBUTTONDBLCLK-Botschaft 470
WM_LBUTTONDOWN-Botschaft 471
WM_LBUTTONUP-Botschaft 471
WM_MBUTTONDBLCLK-Botschaft 472
WM_MBUTTONDOWN-Botschaft 473
WM_MBUTTONUP-Botschaft 473
WM_MDIACTIVATE-Botschaft 474
WM_MDICASCADE-Botschaft 475
WM_MDICREATE-Botschaft 475
WM_MDIDESTROY-Botschaft 476
WM_MDIGETACTIVE-Botschaft 476
WM_MDIICONARRANGE-Botschaft 476
WM_MDIMAXIMIZE-Botschaft 477
WM_MDINEXT-Botschaft 477
WM_MDIRESTORE-Botschaft 478
WM_MDISETMENU-Botschaft 478
WM_MDITILE-Botschaft 478
WM_MEASUREITEM-Botschaft 479
WM_MENUCHAR-Botschaft 479
WM_MENUSELECT-Botschaft 480
WM_MOUSEACTIVATE-Botschaft 481
WM_MOUSEMOVE-Botschaft 481
WM_MOVE-Botschaft 482
WM_NCACTIVATE-Botschaft 482
WM_NCCALCSIZE-Botschaft 483
WM_NCCREATE-Botschaft 483
WM_NCDESTROY-Botschaft 483
WM_NCHITTEST-Botschaft 484
WM_NCLBUTTONDBLCLK-Botschaft 485
WM_NCLBUTTONDOWN-Botschaft 485
WM_NCLBUTTONUP-Botschaft 486
WM_NCMBUTTONDBLCLK-Botschaft 486
WM_NCMBUTTONDOWN-Botschaft 487
WM_NCMBUTTONUP-Botschaft 488
WM_NCMOUSEMOVE-Botschaft 488
WM_NCPAINT-Botschaft 489
WM_NCRBUTTONDBLCLK-Botschaft 489
WM_NCRBUTTONDOWN-Botschaft 490

WM_NCRBUTTONUP-Botschaft 490
WM_NEXTDLGCTL-Botschaft 491
WM_PAINT-Botschaft 491
WM_PAINTCLIPBOARD-Botschaft 446, 492
WM_PAINTICON-Botschaft 492
WM_PALETTECHANGED-Botschaft 493
WM_PARENTNOTIFY-Botschaft 493
WM_PASTE-Botschaft 494
WM_QUERYDRAGICON-Botschaft 494
WM_QUERYENDSESSION-Botschaft 460, 495
WM_QUERYNEWPALETTE-Botschaft 495
WM_QUERYOPEN-Botschaft 495
WM_QUIT-Botschaft 496
WM_RBUTTONDBLCLK-Botschaft 496
WM_RBUTTONDOWN-Botschaft 497
WM_RBUTTONUP-Botschaft 497
WM_RENDERALLFORMATS-Botschaft 498
WM_RENDERFORMAT-Botschaft 498
WM_SETCURSOR-Botschaft 499
WM_SETFOCUS-Botschaft 499
WM_SETFONT-Botschaft 466, 500
WM_SETREDRAW-Botschaft 500
WM_SETTEXT-Botschaft 427, 501
WM_SHOWWINDOW-Botschaft 501
WM_SIZE-Botschaft 502
WM_SIZECLIPBOARD-Botschaft 502
WM_SPOOLERSTATUS-Botschaft 503
WM_SYSCHAR-Botschaft 447, 503
WM_SYSCOLORCHANGE-Botschaft 504
WM_SYSCOMMAND-Botschaft 449, 505
WM_SYSDEADCHAR-Botschaft 506
WM_SYSKEYDOWN-Botschaft 447, 506
WM_SYSKEYUP-Botschaft 447, 507
WM_TIMECHANGE-Botschaft 509
WM_TIMER-Botschaft 509
WM_UNDO-Botschaft 509
WM_VKEYTOITEM-Botschaft 510
WM_VSCROLL-Botschaft 510
WM_VSCROLLCLIPBOARD-Botschaft 511
WM_WININICHANGE-Botschaft 511
WNDCLASS-Typ 545
WORD
  Werte verketten 280
WORD-Typ 546
WriteComm-Funktion 409
WritePrivateProfileString-Funktion 410
WriteProfileString-Funktion 411
WS_EX_XXXX-Konstanten 625
WS_HSCROLL-Stil 438, 443, 465, 622
WS_XXXX-Konstanten 622 f.
wvsprintf-Funktion 414

Y
Yield-Funktion 415

Z
Zeichen
  Abstand 361
  alphabetische 250
  alphanumerische 250
  Botschaften 396
  Breite 152
  empfangen 398
  großgeschriebene 251
  in Puffer formatieren 414
  kleingeschriebene 251
  konvertieren 290 f., 404
  Schnittstelle 398
  tote 456, 506
  übertragen 397
  von Schnittstelle einlesen 308
Zeichenabstands-Konstanten 625
Zeichenobjekte
  vordefinierte 620
Zeichensatz-Konstanten 625
Zeichensätze
  ändern 51 f.
Zeichenwerkzeuge
  zurücksetzen 400
zeichnen
  aktueller Modus 199
  beenden 124
  besitzergezeichnete Steuerelemente 594
  Fenster 56
  Funktionen 12 f., 40
  Modus 352
  Tools 519
  Werkzeuge 400, 620
Zeiger
  Integer 543
  konvertieren 280
Zeit
  seit Systemstart 163
zusätzliche Bytes
  Klasse 546
Zwischenablage
  Besitzer 157
  Bildlauf 465
  Bildlaufleisten 511
  Daten 332 f., 498
  Daten abrufen 156
  Daten-Handle 332 f.

Format 446
Formate auflisten 126
Formate registrieren 311
Formatnamen 157
Formatrangfolge 193
Funktionen 41
Größe 502
Inhalt 492, 494
Kette 61
leeren 121, 458
neu zeichnen 465
öffnen 293
schließen 65
verfügbare Formate 68, 252
Viewer-Kette 158, 334, 447, 459
Viewer-Kette ändern 61
zeichnen 459

REFERENZ-
HANDBUCH

OBJECT WINDOWS® C++

BORLAND

BORLAND GMBH, GAUTINGER STR. 10,POSTFACH 1209, D-8130 STARNBERG, TELEFON 0 81 51 / 263-0, TELEFAX 0 81 51 / 142 09
CORPORATE HEADQUARTERS: 1800 GREEN HILLS ROAD, P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001. (408) 438-530
OFFICES IN: AUSTRALIA, DENMARK, FRANCE, GERMANY, ITALY, JAPAN, NEW ZEALAND, SINGAPORE, SWEDEN AND THE UNITED KINDOM * ARTIKEL-NR. 135300